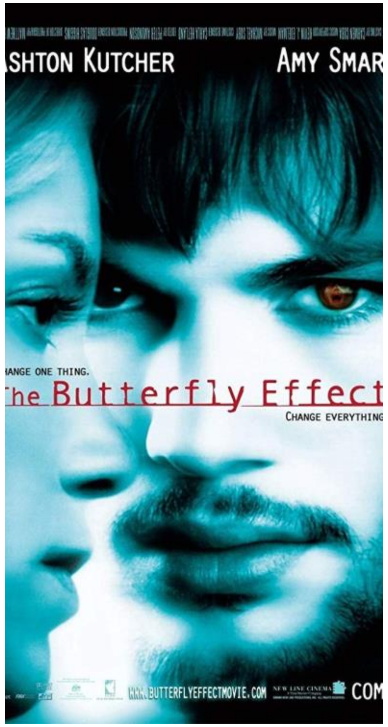
Unification of different types of ML Problems

Dr. Fayyaz Minhas

Department of Computer Science
University of Warwick
https://warwick.ac.uk/fac/sci/dcs/teaching/material/cs909/

Common Theme



- Objective
  - Maximize Generalization
    - Expected predictive quality over unseen/novel test data
    - Minimize expected risk

- Structural Risk Minimization
  - Loss function
    - Limits training error
      - Promotes learning from training data
  - Regularization
    - Doesn’t allow small (or unrelated) changes in input produce large changes in the output
    - Controls complexity of the boundary of the classifier
    - Capacity Control Term (Limits the possibility of memorization)

- Strongly Recommended “Watch” *Complete Statistical Theory of Learning by (Vladimir Vapnik)* https://www.youtube.com/watch?v=Ow25mjFjSmg.

Representation: $\mathrm{f}(\boldsymbol{x};\mathbf{w},\mathrm{b}) = \boldsymbol{w}^T\boldsymbol{x} + \mathrm{b}$ or kernelized $\mathrm{f}(\boldsymbol{x};\boldsymbol{\alpha},\mathrm{b}) = \mathrm{b} + \sum_{j=1}^{N}\alpha_j k(\boldsymbol{x},\boldsymbol{x}_j)$ via the Representer Theorem with Structural Risk Minimization under the general form

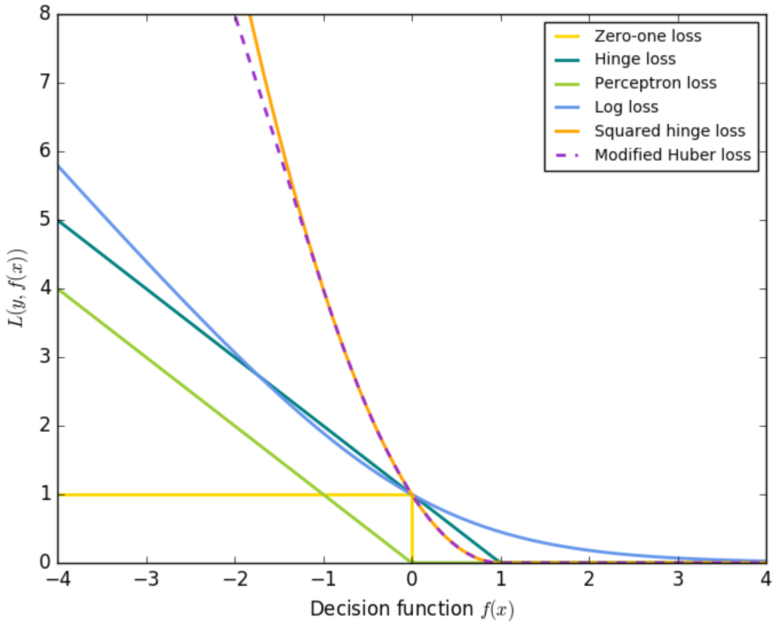
$$\min_{\boldsymbol{w}} \lambda R(\boldsymbol{w}) + E[error\ or\ loss\ over\ training\ examples]$$

$R(\boldsymbol{w})$ is the regularization term and SRM provides a bound on generalization error. The goal is to minimize the expected error but under i.i.d. assumption $\mathrm{E}[\mathrm{loss}] = \frac{1}{N}\sum_{i=1}^{N} l(f(\boldsymbol{x}_i), y_i)$

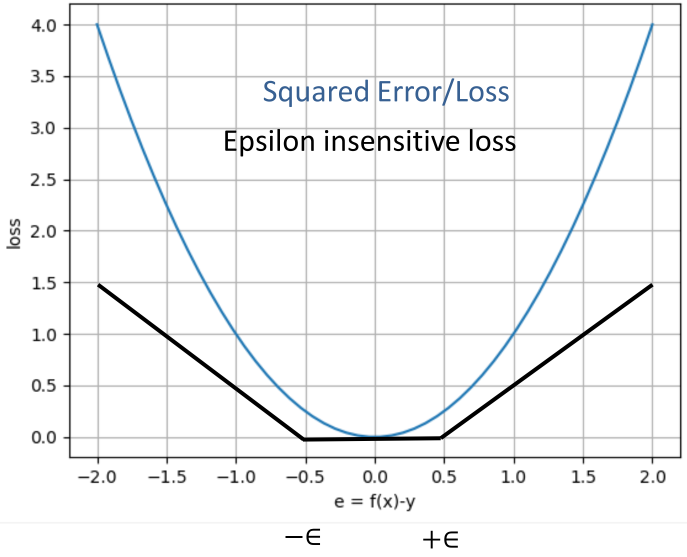
Name	Evaluation (Optimization Problem)	Explanation
Perceptron	$min_w \sum_{i=1}^{N} max(0, 1 - y_i f(\boldsymbol{x};\boldsymbol{w}))$	Uses hinge loss for classification
SVC (Linear)	$min_w \frac{\lambda}{2}\boldsymbol{w}^T\boldsymbol{w} + \sum_{i=1}^{N} max(0, 1 - y_i f(\boldsymbol{x};\boldsymbol{w}))$	Regularized Perceptron
SVC (Kernelized)	$\min_{\alpha,b} \frac{\lambda}{2}\sum_{i,j=1}^{N}\alpha_i\alpha_j k(\boldsymbol{x}_i,\boldsymbol{x}_j) + \frac{1}{N}\sum_{i=1}^{N}\max\left\{0, 1 - y_i\left(b + \sum_{j=1}^{N}\alpha_j k(\boldsymbol{x}_i,\boldsymbol{x}_j)\right)\right\}$	Kernelized SVC
Logistic Regression	$\min_{\boldsymbol{w},b} \frac{1}{2}\lVert\boldsymbol{w}\rVert^2 + \frac{C}{\boldsymbol{N}}\sum_{\boldsymbol{i}=\boldsymbol{1}}^{\boldsymbol{N}}\log(\exp(-y_i f(x_i)) + 1)$	Uses the logistic loss for classification.
PCA	$\min_{\boldsymbol{w}} \boldsymbol{\lambda}\boldsymbol{w}^T\boldsymbol{w} + (\boldsymbol{V} - \boldsymbol{w}^T\boldsymbol{C}\boldsymbol{w})$	Find (orthogonal) direction(s) by minimizing the loss in variance after projection
OLS	$\min_{w} \sum_{i=1}^{N}(\boldsymbol{w}^T\boldsymbol{x}_i - y_i)^2 = \lVert\boldsymbol{X}\boldsymbol{w} - \boldsymbol{y}\rVert^2$	Find best linear regression fit under squared loss
SVR (Linear)	$\min_{\boldsymbol{w},b} \frac{1}{2}\boldsymbol{w}^T\boldsymbol{w} + \frac{C}{\boldsymbol{N}}\sum_{\boldsymbol{i}=\boldsymbol{1}}^{\boldsymbol{N}} max(0, \lvert f(\boldsymbol{x}_i) - y_i\rvert - \epsilon)$	Uses epsilon-insensitive loss for regression
SVR (Kernelized)	$\min_{\alpha,b} \frac{1}{2}\sum_{i,j=1}^{N}\alpha_i\alpha_j\, k(\boldsymbol{x}_i,\boldsymbol{x}_j) + \frac{C}{\boldsymbol{N}}\sum_{\boldsymbol{i}=\boldsymbol{1}}^{\boldsymbol{N}}\boldsymbol{max}\left(0, \left\lvert\sum_{j=1}^{N} k(\boldsymbol{x}_i,\boldsymbol{x}_j) + b - y_i\right\rvert - \epsilon\right)$	Kernelized form of the above
Ridge Regression	$\min_{\boldsymbol{w},b} \alpha\lVert\boldsymbol{w}\rVert^2 + \lVert\boldsymbol{X}\boldsymbol{w} - \boldsymbol{y}\rVert^2$	OLS with regularization (squared norm)
Lasso	$\min_{\boldsymbol{w},b} \alpha\lVert\boldsymbol{w}\rVert_1 + \lVert\boldsymbol{X}\boldsymbol{w} - \boldsymbol{y}\rVert^2$	Use 1-norm regularization (minimize sum of absolute values rather than their squares)
Elastic Net	$\min_{\boldsymbol{w},b} \alpha\rho\lVert\boldsymbol{w}\rVert_1 + \frac{\alpha(1-\rho)}{2}\lVert\boldsymbol{w}\rVert^2 + \lVert\boldsymbol{X}\boldsymbol{w} - \boldsymbol{y}\rVert^2$	Uses both types of regularization
Huber Regressor	$\min_{w,b} \alpha\lVert\boldsymbol{w}\rVert^2 + \sum_{i=1}^{N} l_{huber}(f(x_i, y_i))$ with $l_{huber}(f(x_i, y_i)) = \begin{cases}\frac{1}{2}(y - f(x))^2 & if\ \lvert y - f(x)\rvert < \delta \\ \delta(\lvert y - f(x)\rvert - \frac{1}{2}\delta) & else\end{cases}$	Used for robust regression as huber loss is less sensitive to outliers than squared loss

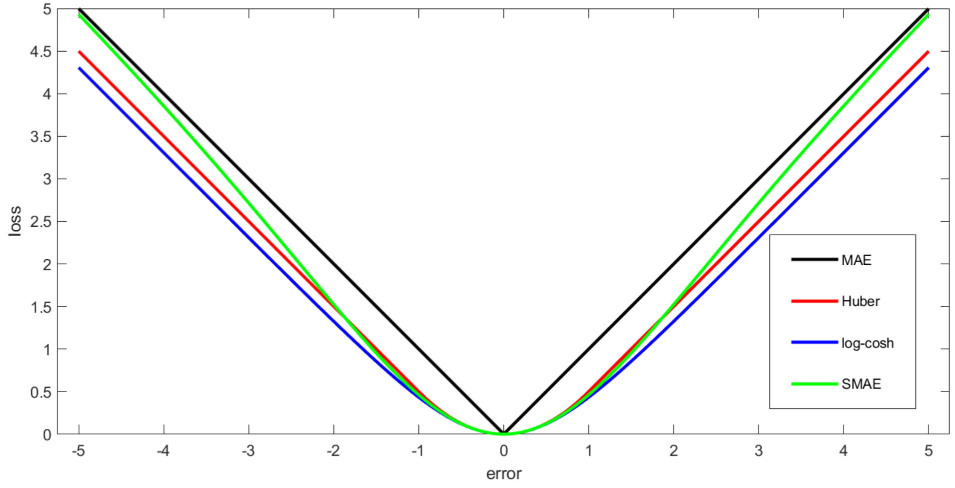
Coding: https://scikit-learn.org/stable/modules/linear_model.html

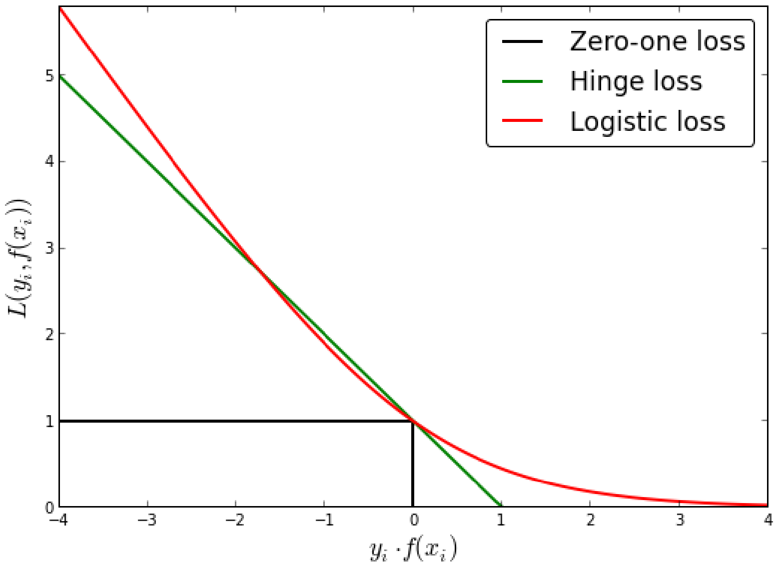
Loss Functions: $l(f(x_i, y_i)$



- Quantify Error
  - Misclassification
  - Misregression
  - Misreconstruction
  - Misclustering, Misranking, Misretrieval, ….
- The loss function determines the behaviour of the predictor
- More importantly, it determines the type of ML problem being solved
- Loss functions on the previous slide are all convex losses
  - Guaranteed single minima and convergence through gradient descent
  - Some even lead to closed form optimization which is great
  - However: LeCun, Yann. "[Who is afraid of non-convex loss functions]()." *NIPS Workshop on Efficient Machine Learning*. 2007.
- A loss function doesn't even have to operate at a per-example level







Regularization

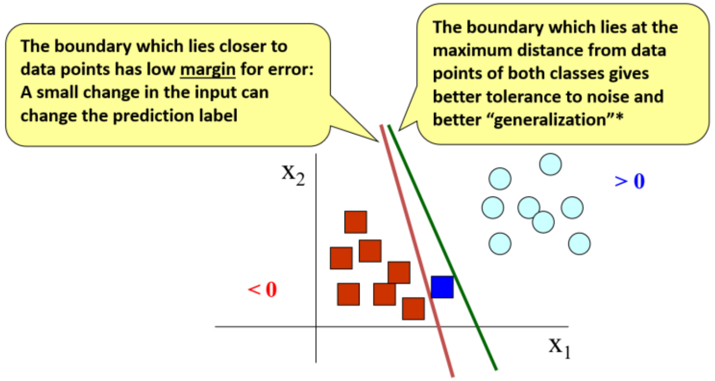
- Small changes in input should produce small changes in output
  – Achieved by minimization of the norm of the weight vector

$$R(\boldsymbol{w}) = \|\boldsymbol{w}\|_2^2 = w_1^2 + w_2^2 + \cdots + w_d^2$$

- In general

$$\|\boldsymbol{w}\|_p = (|w_1|^p + |w_2|^p + \cdots + |w_d|^p)^{1/p}$$
$$\|\boldsymbol{w}\|_1 = |w_1| + |w_2| + \cdots + |w_d|$$
$$\|\boldsymbol{w}\|_0 = number\ of\ non - zero\ vector\ elements$$

- Enables generalization esp. when the number of data points is quite small in comparison to the number of dimensions of each data point: A cure to the Curse of dimensionality
  – Given only training examples, optimizing empirical error over only a small number of training examples can lead to models that do not generalize to unseen examples effectively



Small weights limit "the butterfly effect"

- *Let's quantify how sensitive the model is to a perturbation of its input*

- $\boldsymbol{f}(\boldsymbol{x}) = \boldsymbol{w}^{\boldsymbol{T}}\boldsymbol{x} + b$
- $\boldsymbol{f}(\boldsymbol{x} + \boldsymbol{\delta x}) = \boldsymbol{w}^{\boldsymbol{T}}(\boldsymbol{x} + \boldsymbol{\delta x}) + \boldsymbol{b} = \boldsymbol{w}^{\boldsymbol{T}}\boldsymbol{x} + \boldsymbol{b} + \boldsymbol{w}^{\boldsymbol{T}}\boldsymbol{\delta x} = \boldsymbol{f}(\boldsymbol{x}) + \boldsymbol{w}^{\boldsymbol{T}}\boldsymbol{\delta x}$
- $\boldsymbol{f}(\boldsymbol{x} + \boldsymbol{\delta x}) - \boldsymbol{f}(\boldsymbol{x}) = \boldsymbol{w}^{\boldsymbol{T}}\boldsymbol{\delta x}$
- $\|\boldsymbol{f}(\boldsymbol{x} + \boldsymbol{\delta x}) - \boldsymbol{f}(\boldsymbol{x})\| = \|\boldsymbol{w}^{\boldsymbol{T}}\boldsymbol{\delta x}\| \leq \|\boldsymbol{w}\|\|\boldsymbol{\delta x}\|$ **(using Cauchy-Schwarz inequality)**

- **Therefore,** $\frac{\|\boldsymbol{f}(\boldsymbol{x}+\boldsymbol{\delta x}) - \boldsymbol{f}(\boldsymbol{x})\|}{\|\boldsymbol{\delta x}\|} \leq \|\boldsymbol{w}\|$

Change in model output per unit additive change in input is upper bounded by $\|\boldsymbol{w}\|$.

Consequently, minimizing the norm of the weight vector (or its square) would lead to a regularization effect as it would limit the effect of any change in the input on the output.

Vapnik showed that **minimizing "structural risk"** (combination of empirical error over training examples and the norm of the weight vector) **leads to minimization of the upper bound on generalization error over unseen examples effectively achieving a solution to the curse of dimensionality.**

$$R(\boldsymbol{w}) \leq R_{emp}(\boldsymbol{w}) + \Omega\left(\frac{1}{N}, \frac{1}{\|\boldsymbol{w}\|}, d\right)$$

Understanding norm-based Regularization

- Remember, output is a weighted combination of the input

$$f(\boldsymbol{x}) = \boldsymbol{w}^T\boldsymbol{x}$$

- Minimizing $||\boldsymbol{w}||_p$
  - p = 2: pulls the point towards the origin
    - Reduces the length of the weight vector and prevents them from growing larger
  - p = 1: reduces the axes coordinates individually
    - Reduce the magnitude of individual weights
      - Smaller individual feature components
        - » Sparse solutions (i.e., fewer non-zero weights than with p = 2)
      - Used for reducing or selecting features to only important ones
  - p = 0: reduces the number of non-zero components
    - Reduce the number of “active” features
    - Feature selection
    - Difficulties in optimization

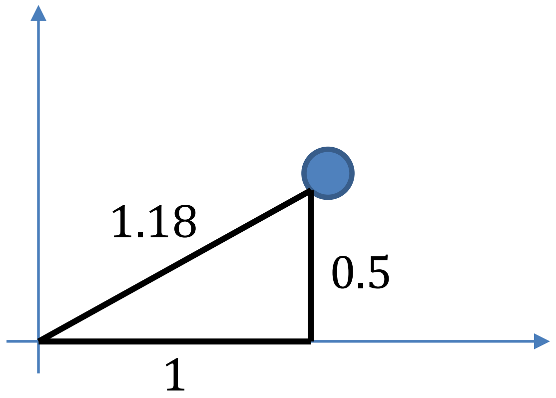
Assume a vector

$$\boldsymbol{w} = \begin{bmatrix} 1 \\ 0.5 \end{bmatrix}$$

$$||\boldsymbol{w}||_{p=2} = \sqrt{1 + 0.25} = 1.18$$

$$||\boldsymbol{w}||_{p=1} = 1 + 0.5 = 1.5$$

$$||\boldsymbol{w}||_{p=0} = 2$$



Regularization: Not limited to norm-based regularization

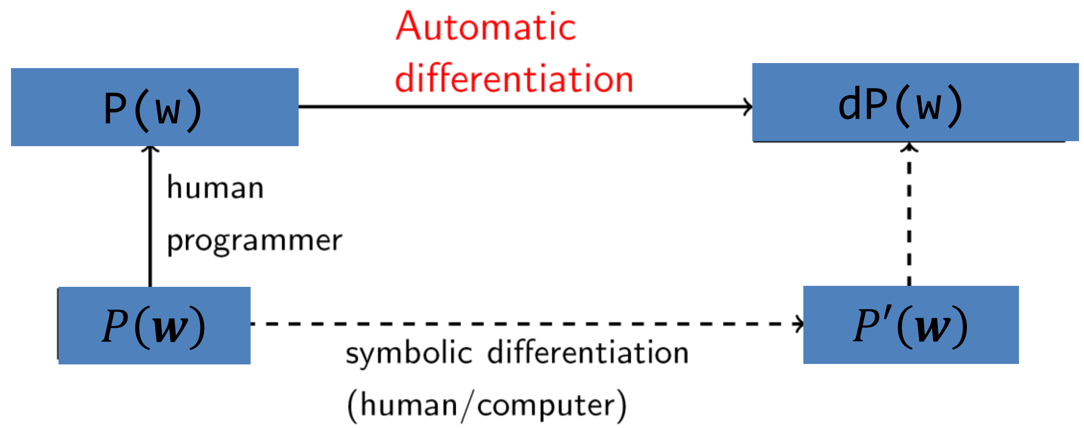
- Small changes in input should produce small changes in output
- More accurately, what we want is
  - “Non-causal” changes in input should not change the output
    - Changes that are not causally linked to label assignment
    - For example:
      - If our goal is to classify horses vs unicorns: A rotated horse is still a horse
      - However, if a change to the horse makes it a unicorn, the ML model prediction should change
    - The ML model should be “Invariant” to such changes
- This idea forms the basis of a number of different type of approaches that have a regularization effect
  - Data augmentation
  - Adversarial Training Perturbations
  - Contrastive learning
  - Drop-out
  - Invariant Risk Minimization
  - Learning using statistical Invariants

How to program any ML model?

- If you can define a loss function
- And a regularizer
- The rest can be automated For any ML problem*!
  - Using Automatic Differentiation Libraries
    - Autograd
    - PyTorch
    - TensorFlow
    - JAX
    - Zygote.jl

REO and SRM are all you need!

- **Representation**
  - *How does the model produce its output given its input*
    - $\boldsymbol{f}(\boldsymbol{x};\boldsymbol{w}) = \boldsymbol{w}^T\boldsymbol{x}$
- **Evaluation (SRM/Definition of Optimization Problem)**
  - Define a loss function and a regularization strategy write the optimization problem
  - $min_{\boldsymbol{w}} P(\boldsymbol{w};\boldsymbol{X},\boldsymbol{y}) = \frac{\lambda}{2}\boldsymbol{w}^T\boldsymbol{w} + \sum_{i=1}^{N} max(0, 1 - y_i f(\boldsymbol{x};\boldsymbol{w}))$
- **Optimization**
  - Obtain gradient $\nabla_w P(w) = \frac{\partial P(w)}{\partial x}$ through an automatic differentiation method
  - Apply gradient descent (or other optimization) updates until convergence
    - $w \leftarrow w - \alpha \nabla_w P(w)$
  - **Successful optimization is necessary for generalization (but not sufficient). Must check for successful optimization!**



Go through this exercise:
https://github.com/foxtrotmike/CS909/blob/master/barebones.ipynb

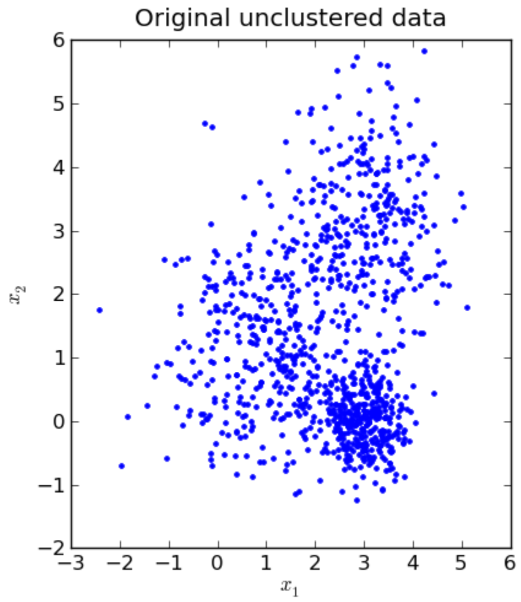
*Terms and conditions apply

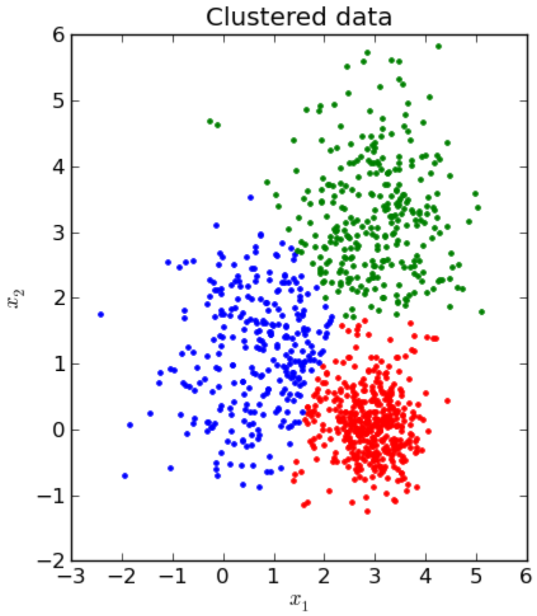
An exercise into SRM

CLUSTERING

Clustering

- Input
  - Typically, a Data Matrix (X)
  - Unsupervised technique
- Grouping objects such that:
  - Objects within the same group (cluster) are more similar to each other and different from objects in other groups
    - This is the underlying optimization problem of all clustering methods
- Metrics
  - Using (unlabeled) training data itself
    - Davies-Bouldin Index
    - Dunn's Index
    - Silhouette Coefficient
  - Using external (test data) with cluster assignments by experts as ground truth
    - Purity
    - Jaccard Index
    - Dice Index





Read: https://en.wikipedia.org/wiki/Cluster_analysis

k-means Clustering

Most commonly used algorithm for clustering

Input: Data Matrix **X**

Hyper-parameter: Number of clusters, Initial Cluster Centers

Output:

Assignment of each example to a cluster center

Initialize $\boldsymbol{m}_i, i = 1, \ldots, k$, for example, to k random $\boldsymbol{x}^t$
Repeat
For all $\boldsymbol{x}^t \in \mathcal{X}$

$$b_i^t \leftarrow \begin{cases} 1 & \text{if } \|\boldsymbol{x}^t - \boldsymbol{m}_i\| = \min_j \|\boldsymbol{x}^t - \boldsymbol{m}_j\| \\ 0 & \text{otherwise} \end{cases}$$

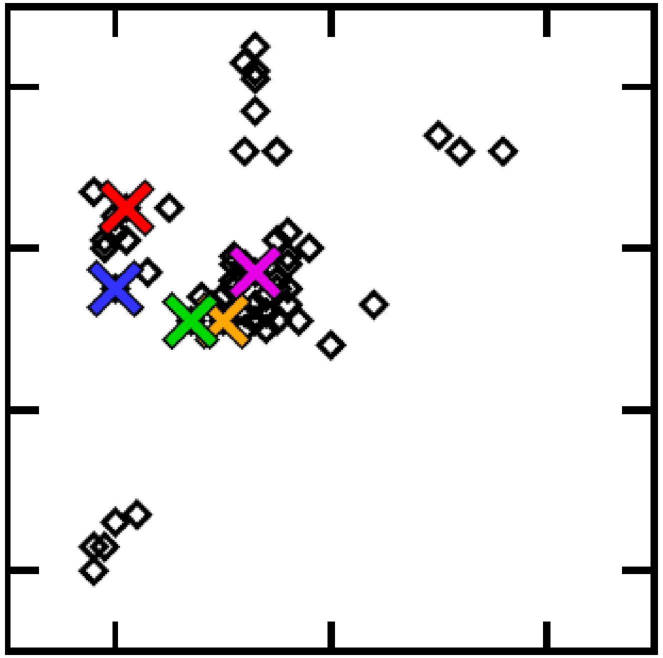
For all $\boldsymbol{m}_i, i = 1, \ldots, k$

$$\boldsymbol{m}_i \leftarrow \sum_t b_i^t \boldsymbol{x}^t / \sum_t b_i^t$$

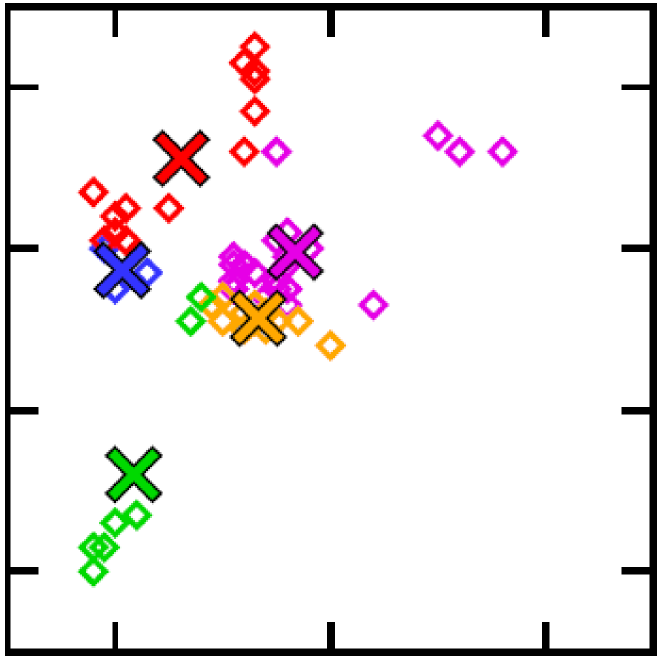
Until $\boldsymbol{m}_i$ converge

- b_i^t is 1 when the i^{th} center is the one closest to x^t

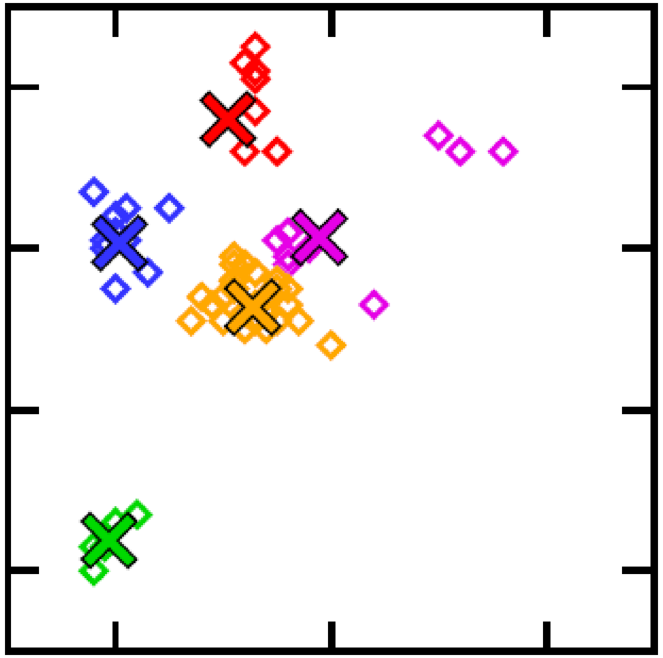
k-means Clustering



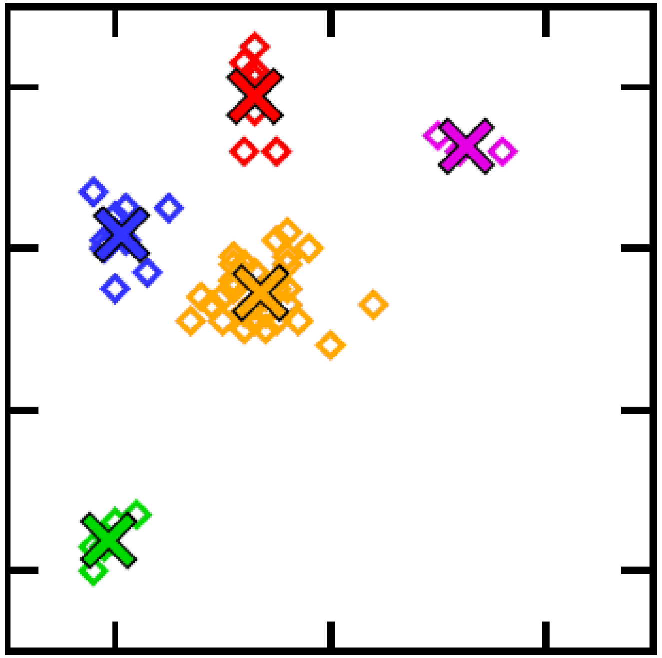
k-means Clustering



k-means Clustering



k-means Clustering



REO for k-means

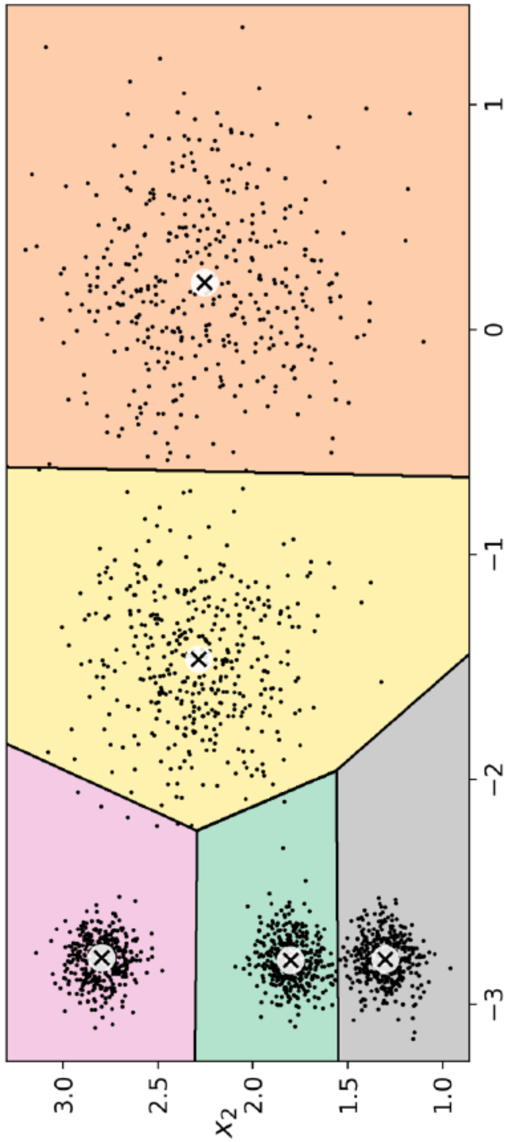
- Representation
  - An example $\boldsymbol{x}$ is assigned a cluster based on its closest "centroid"
    - The K centroids are denoted as: $\boldsymbol{M} = \{\boldsymbol{m_1}, \dots, \boldsymbol{m_K}\}$
    - The cluster assignment for an example based on a distance metric $d(\cdot,\cdot)$ is given by the nearest neighbor rule

$$c(\boldsymbol{x}) = argmin_{j=\{1\dots K\}} d(\boldsymbol{x}, \boldsymbol{m}_j), c(\boldsymbol{x}) \in \{1 \dots K\}$$

- Evaluation
  - We would like to determine the cluster centroids $\boldsymbol{M} = \{\boldsymbol{m_1}, \dots, \boldsymbol{m_k}\}$ and the assignments of training examples to clusters (non-overlapping sets) $\boldsymbol{S} = \{\boldsymbol{S_1}, \dots, \boldsymbol{S_K}\}$ such that the within-cluster distance from centroids is minimized.

$$\min_S \sum_{j=1}^{K} \sum_{x \in S_j} d(\boldsymbol{x}, \boldsymbol{m_j}) \text{ with } \boldsymbol{m_j} = \frac{\boldsymbol{1}}{|\boldsymbol{S_j}|} \sum_{\boldsymbol{x} \in \boldsymbol{S_j}} \boldsymbol{x}$$

- Optimization: This is an NP-Hard problem but the Previously described approximate algorithm leads to a good local optimum
- Hidden Hyperparameters
  - Distance metric: You can get different clustering based on the distance you use
- Regularization?
  - Cluster assignment of an example should not change within a short distance: The choice of your distance metric controls regularization

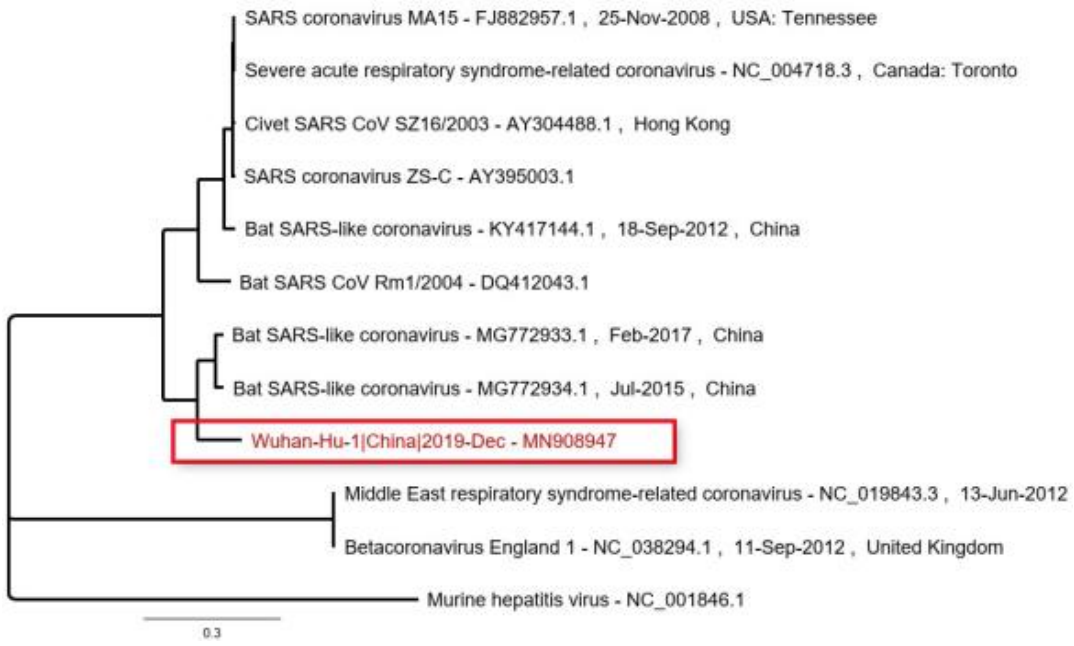


Must read:
https://en.wikipedia.org/wiki/K-means_clustering

Hierarchical Clustering

- Build a hierarchy of clusters
  - Bottom up: Agglomerative Clustering
  - Top down: Divisive clustering
- Allows us to represent the findings in a tree
- We can cutoff at any height to get different number of clusters
- Hyperparameters
  - Distance metric
  - Linkage: How do we define a distance between two sets of points
    - Average, Min, Max, etc…
    - Changes clustering
- Single Dimensional Example showing the dendrogram

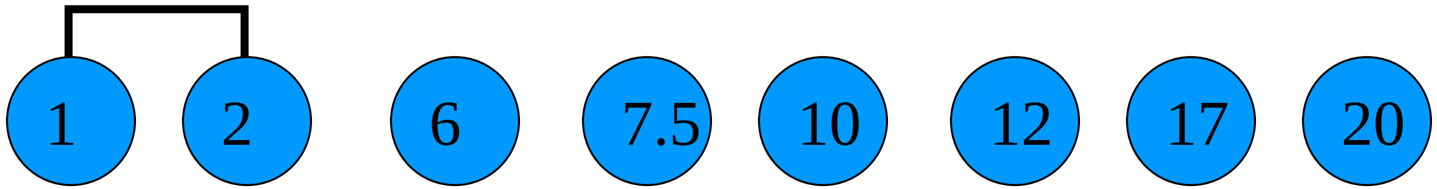
A "Phylogenetic" tree based on genomic distance for SARSCov-2

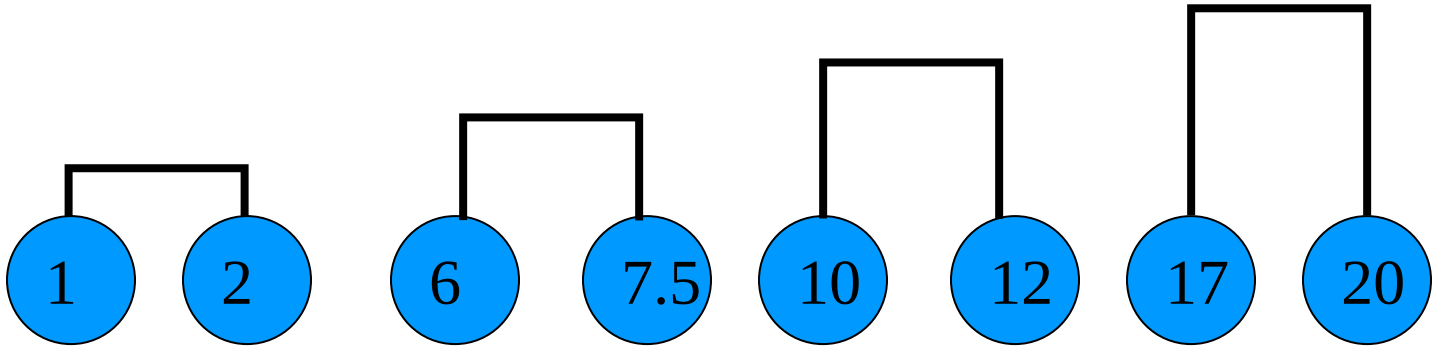


https://nextstrain.org/ncov/global

https://ncbiinsights.ncbi.nlm.nih.gov/2020/01/13/novel-coronavirus/

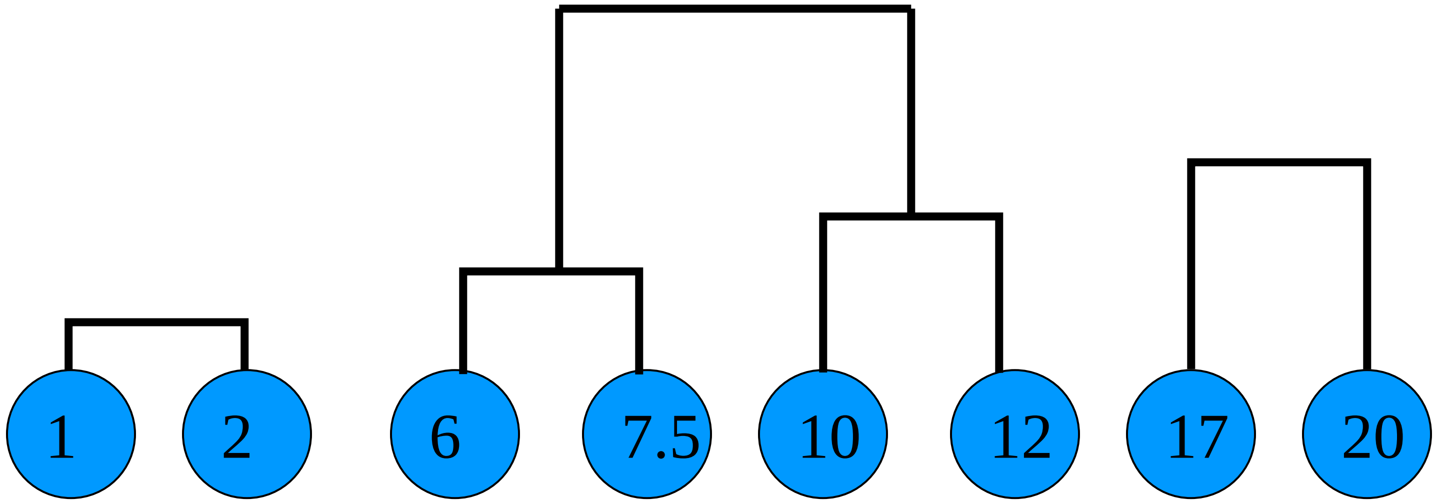
https://en.wikipedia.org/wiki/Hierarchical_clustering


1
2
6
7.5
10
12
17
20

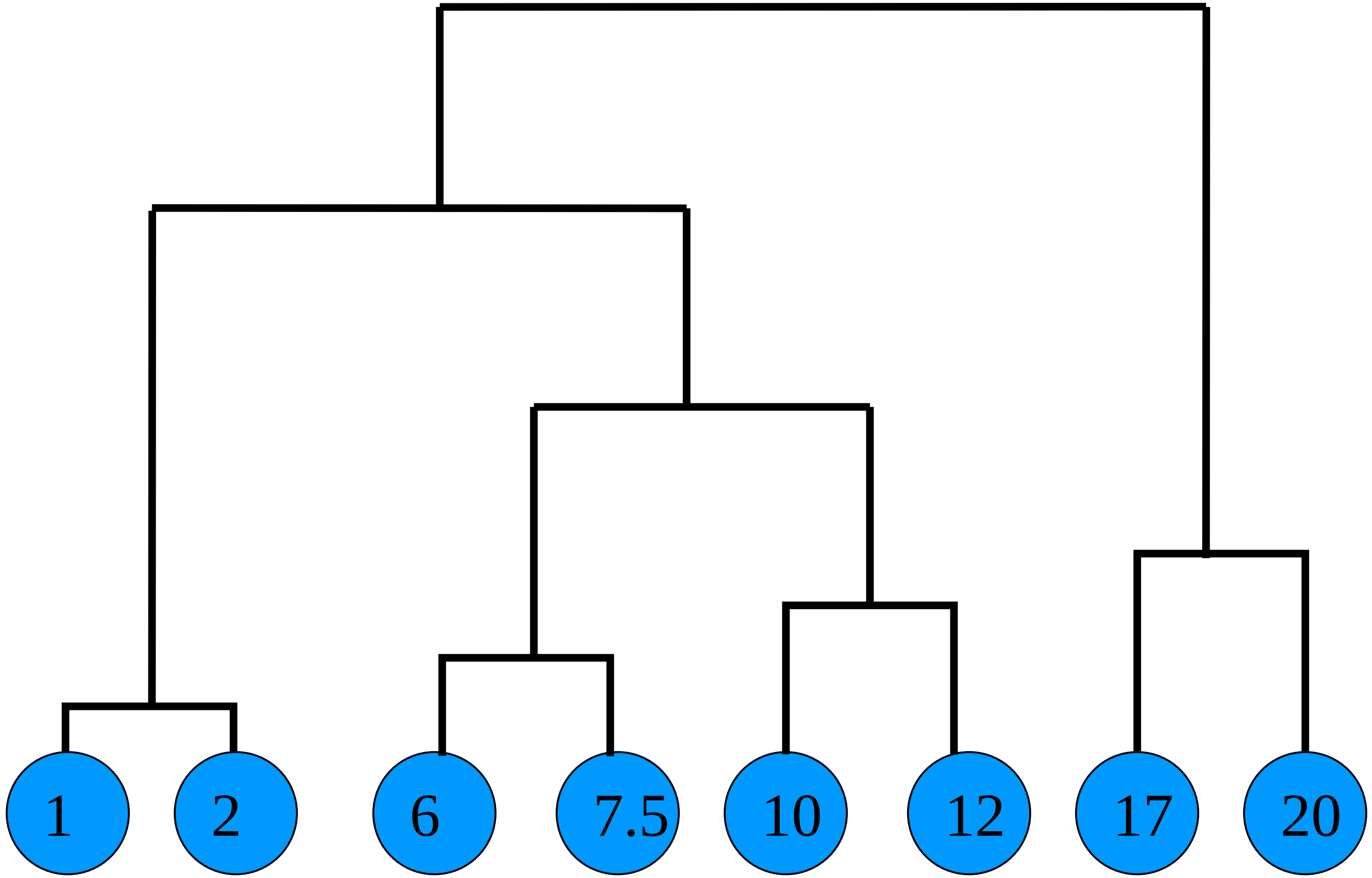

1
2
6
7.5
10
12
17
20

Linkage

- How do we define the distance between clusters
  - Min
  - Max
  - Average



Linkage

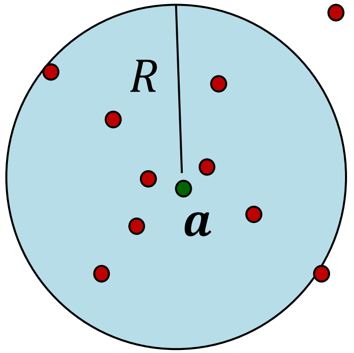


Support Vector Clustering

- Ben-Hur and Vapnik 2001
- No assumptions on the shape and number of clusters
- Enclose all examples in a kernel feature space in a tight sphere centered at "$\boldsymbol{a}$" with radius "R"

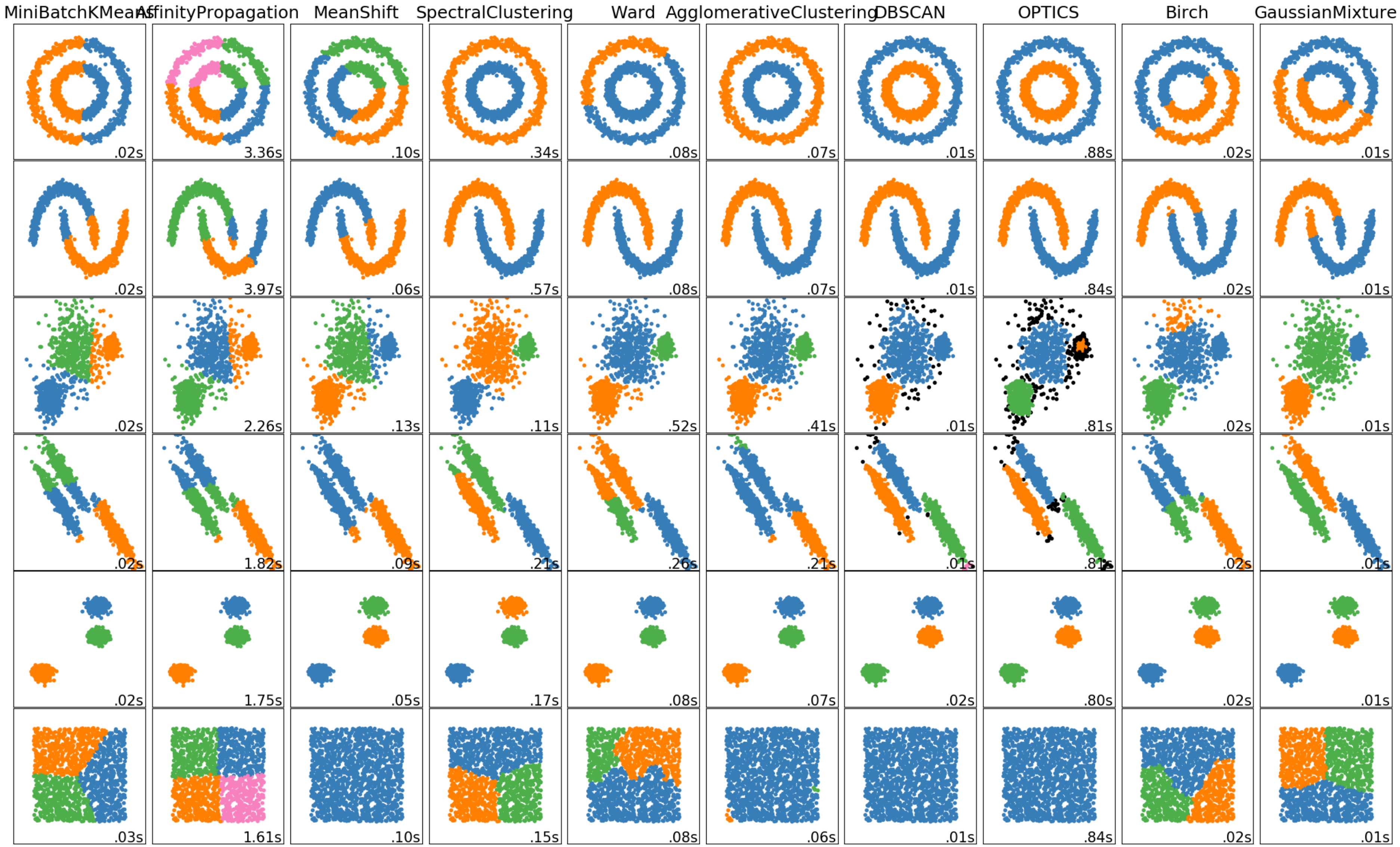
$$min_{R,\boldsymbol{a}}\ R^2 + \mathrm{C}\sum_{i=1}^{N} \max(0, ||\phi(\boldsymbol{x_i}) - \boldsymbol{a}||^2 - R^2\ )$$

- Two points belong to the same cluster if, for all points $\boldsymbol{x}$ in between them $||\phi(\boldsymbol{x}) - \boldsymbol{a}||^2 < R^2$
- Can be kernelized
- No need of specifying the number of clusters a priori



Ben-Hur, Asa, et al. "Support vector clustering." *Journal of machine learning research* 2.Dec (2001): 125-137.

Many Other




```
y_pred = KMeans(n_clusters=3).fit_predict(X)
```

https://scikit-learn.org/stable/auto_examples/cluster/plot_cluster_comparison.html

Method name	Parameters	Scalability	Usecase	Geometry (metric used)
K-Means	number of clusters	Very large n_samples, medium n_clusters with MiniBatch code	General-purpose, even cluster size, flat geometry, not too many clusters	Distances between points
Affinity propagation	damping, sample preference	Not scalable with n_samples	Many clusters, uneven cluster size, non-flat geometry	Graph distance (e.g. nearest-neighbor graph)
Mean-shift	bandwidth	Not scalable with n_samples	Many clusters, uneven cluster size, non-flat geometry	Distances between points
Spectral clustering	number of clusters	Medium n_samples, small n_clusters	Few clusters, even cluster size, non-flat geometry	Graph distance (e.g. nearest-neighbor graph)
Ward hierarchical clustering	number of clusters or distance threshold	Large n_samples and n_clusters	Many clusters, possibly connectivity constraints	Distances between points
Agglomerative clustering	number of clusters or distance threshold, linkage type, distance	Large n_samples and n_clusters	Many clusters, possibly connectivity constraints, non Euclidean distances	Any pairwise distance
DBSCAN	neighborhood size	Very large n_samples, medium n_clusters	Non-flat geometry, uneven cluster sizes	Distances between nearest points
OPTICS	minimum cluster membership	Very large n_samples, large n_clusters	Non-flat geometry, uneven cluster sizes, variable cluster density	Distances between points
Gaussian mixtures	many	Not scalable	Flat geometry, good for density estimation	Mahalanobis distances to centers
Birch	branching factor, threshold, optional global clusterer.	Large n_clusters and n_samples	Large dataset, outlier removal, data reduction.	Euclidean distance between points

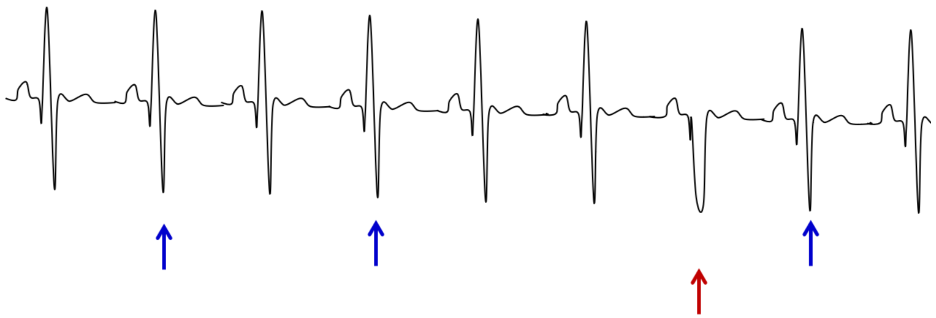
https://scikit-learn.org/stable/auto_examples/cluster/plot_cluster_comparison.html

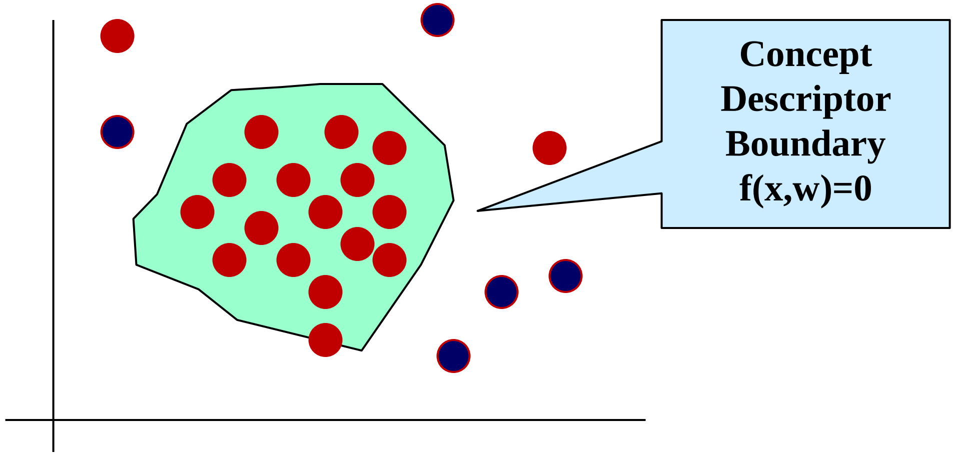
An exercise into SRM

ONE CLASS CLASSIFICATION

One Class Classification

- Unary Classification
- Class Modelling
- Novelty Detection
- Examples for one class only (say normal)
- ***Identify those examples that differ from the given class***

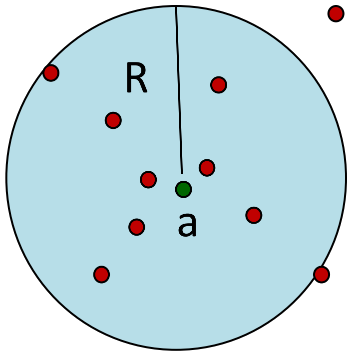




OCC: Support Vector Data Descriptors

- *Finds a hyper-sphere with center at '**a**' and radius **R** so that the target class examples lie within the hyper-sphere*
  - *Error function: Penalize training examples if they lie outside the hypersphere*

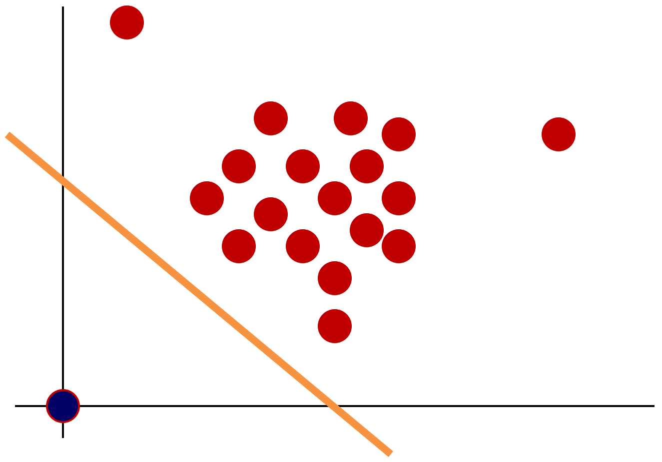
- $\boldsymbol{min}_{\boldsymbol{a},R}\ R^2 + \frac{C}{N}\sum_{i=1}^{N} max(0, \|\boldsymbol{x} - \boldsymbol{a}\|^2 - R^2)$



One-Class SVM (Scholkopf 2001)

- Goal: Separate points of the target class (represented in the kernel space) from the origin with maximum margin
- Representation
  - Linear Separability Case $f(x) = w^T x + b$
    - All given (target) class examples should have $f(x) \geq 0$
    - Consider that the outliers are mapped to the origin $f(0) = b < 0$
- Evaluation
  - Error when
    - A point of the target class produces $f(x) \leq 0$
    - Or the origin is classified as target: $f(0) = w^T x + b = b > 0$
  - Loss function thus becomes: $l(f(\boldsymbol{x}; \boldsymbol{w}, b) = \max(0, -f(x)) + b$
  - Thus: (it can be kernelized)

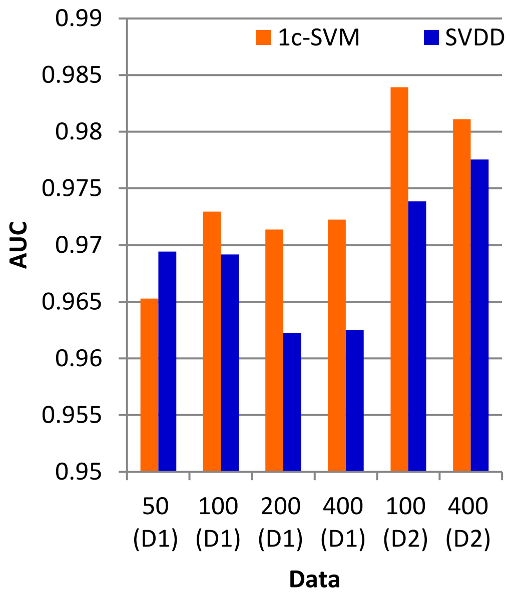
$$\boldsymbol{min}_{\boldsymbol{w},b} \frac{1}{2}\boldsymbol{w}^T\boldsymbol{w} + \frac{C}{N}\sum_{i=1}^{N} max\left(0, -\left(\boldsymbol{w}^T\boldsymbol{x} + b\right)\right) + b$$

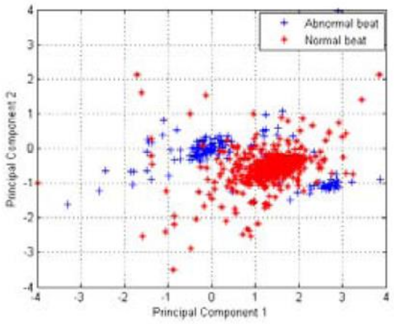


https://scikit-learn.org/stable/modules/generated/sklearn.svm.OneClassSVM.html

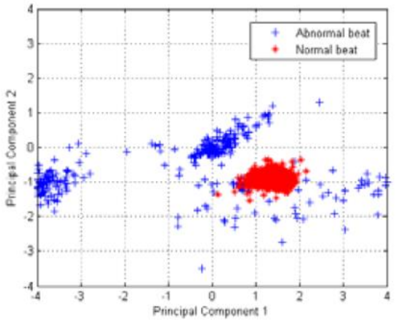
Abnormal Beat Detection in ECG

- ECG based automated detection of abnormal beats has low generalization across individuals
- Solution: Use OCC to train on the normal beats **for each individual**
  - Do not have to give any abnormal beats in training
- Validation on 46 Records from MIT-BIH Database with lead MLII
  - 73,258 normal (~69.0%) and about 32,827 (~31%) abnormal beats

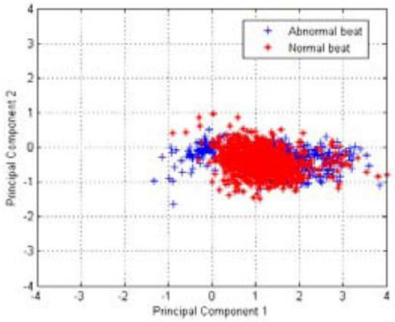




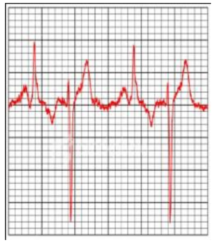
Record 105



Record 200



Record 222



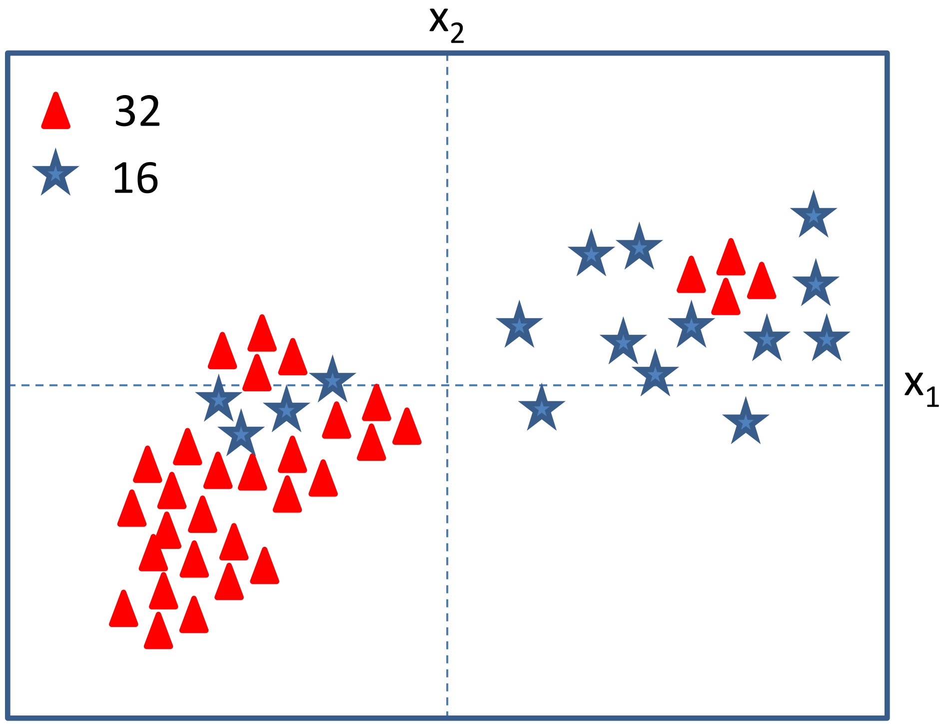
Robust electrocardiogram (ECG) beat classification using discrete wavelet transform. Minhas, F. and Arif, M. 5, 2008, Physiological Measurement, Vol. 29.

An exercise into SRM

TREES, FORESTS AND BOOSTING

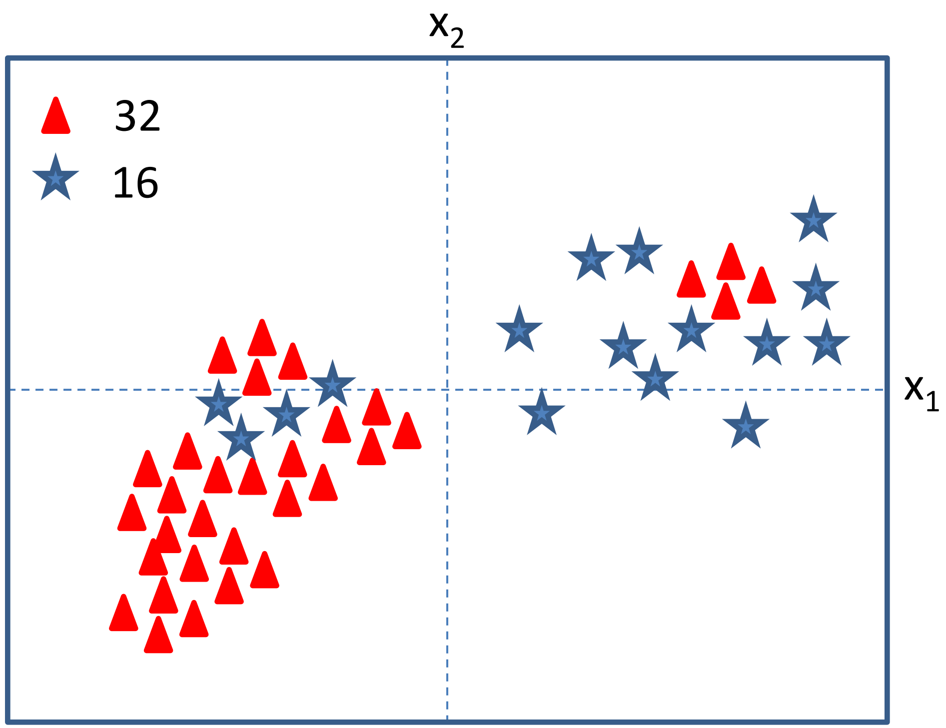
Trees as classifiers

- Assume you have a classification problem



Can we learn a set of rules of assignment of different regions of the feature space to different classes?

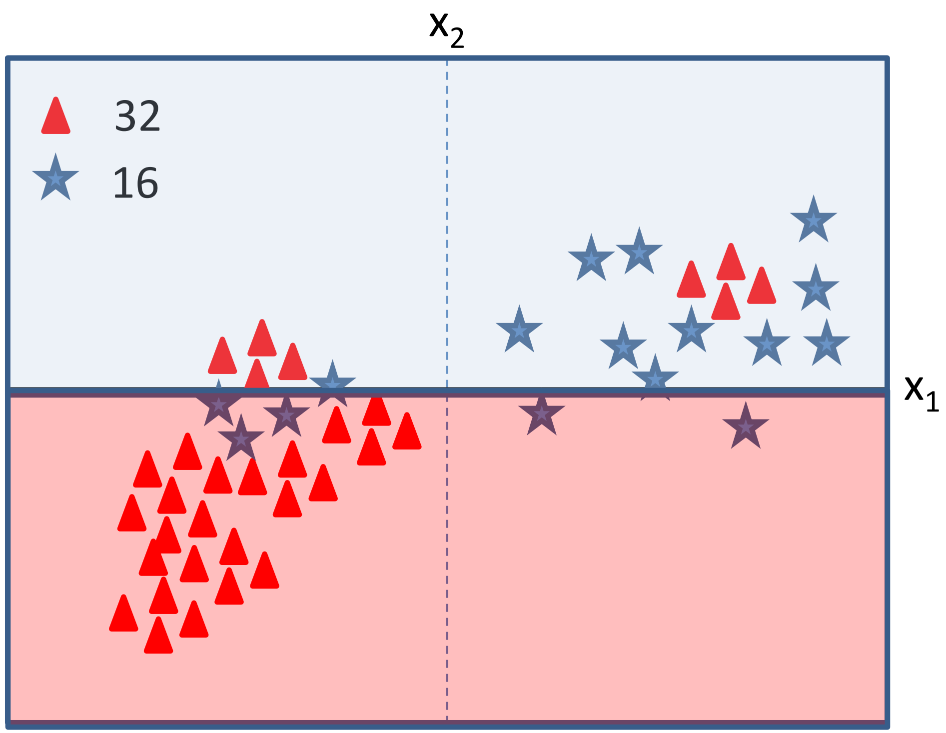
Picking a feature to split



Current Error Rate: 16/48 = 1/3

At each step pick the feature and split that gives the most “information gain” or the most reduction in error

Check x_2



Total points: 48
Current Error Rate: 16/48 = 1/3

For a split along $x_2 = 0$
Total points in the top half = 19 out of 48
Error in the top half: 8/19

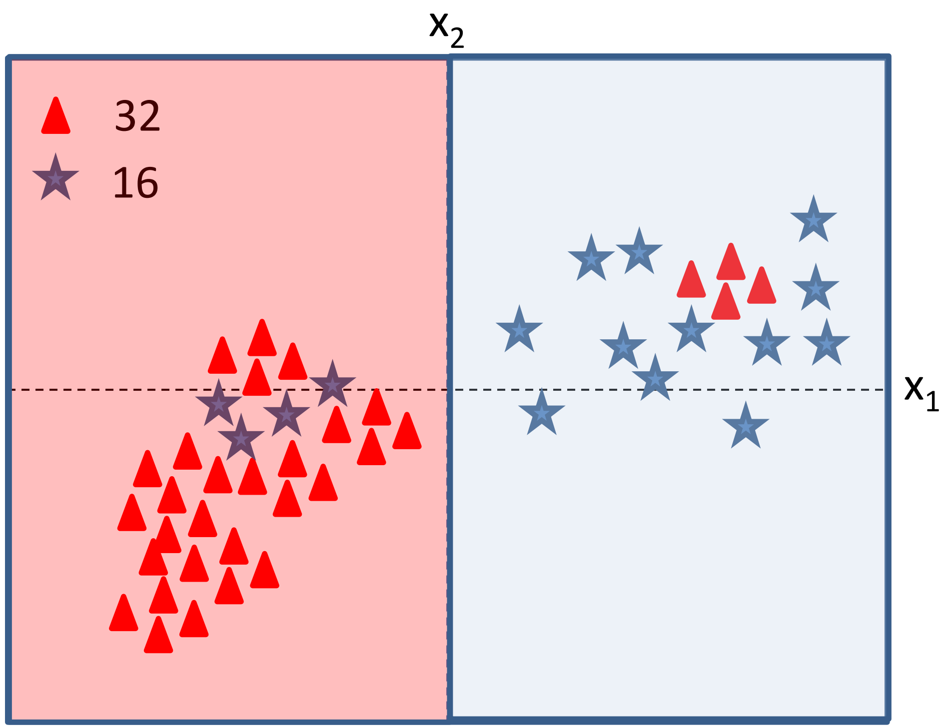
Total points in the bottom half: 29
Error in the bottom half = 5/29

Total error: $\frac{8}{19}\frac{19}{48} + \frac{5}{29}\frac{29}{48} = 13/48$

Reduction in error = 16/48-13/48 = 3/48

At each step pick the feature that gives the most “information gain”

Check x_1



x_1 Gives the most improvement in error rate

Total points: 48
Current Error Rate: 16/48 = 1/3

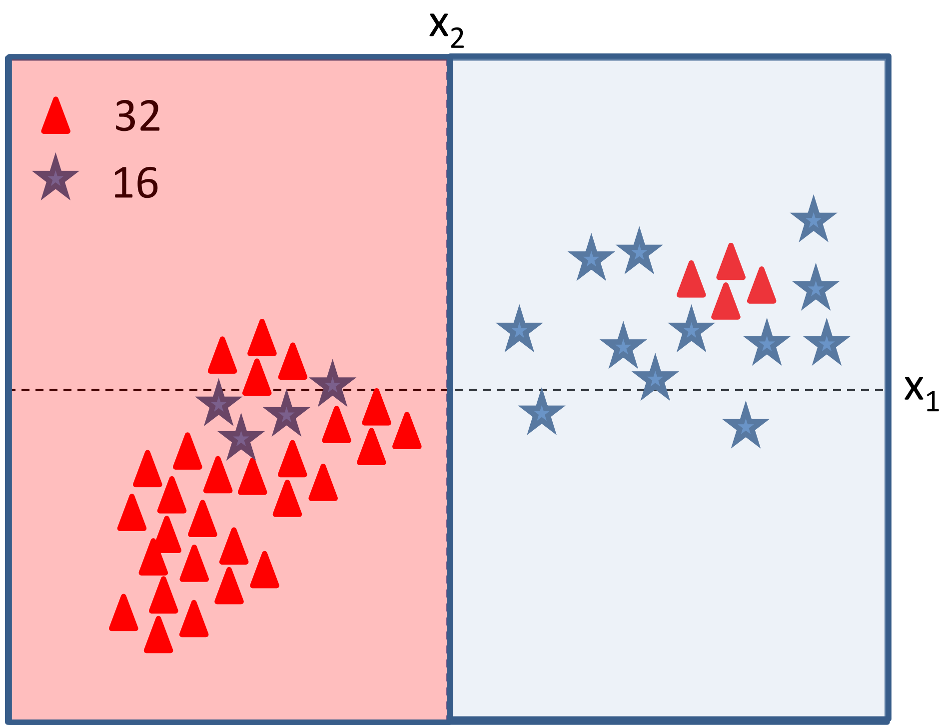
For a split along $x_1 = 0$
Total points in the L half = 32 out of 48
Error in the L half: 4/32

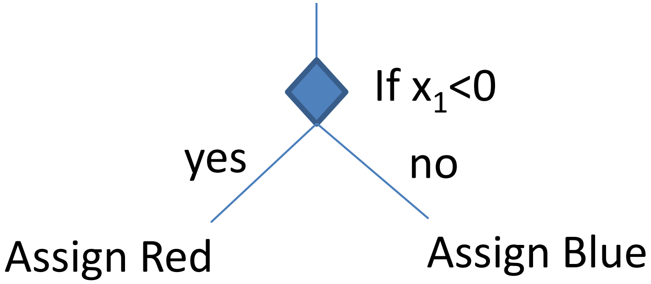
Total points in the R half: 16
Error in the bottom half = 4/16

Total error: $\frac{4}{32}\frac{32}{48} + \frac{4}{16}\frac{16}{48}$ = 8/48

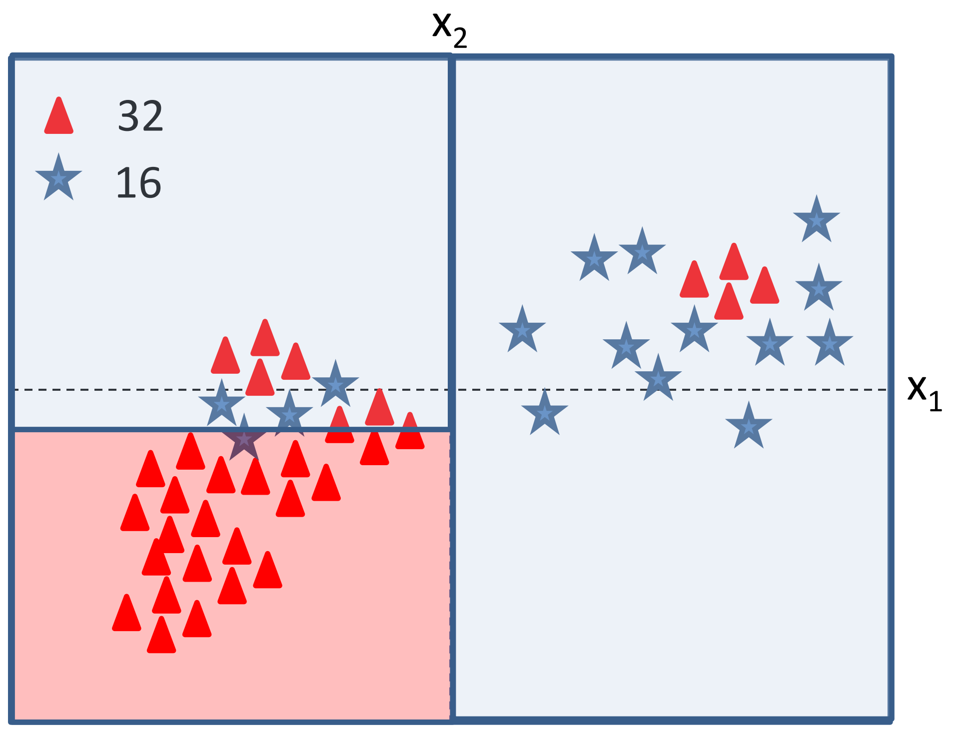
Reduction in error = 16/48-8/48 = 8/48

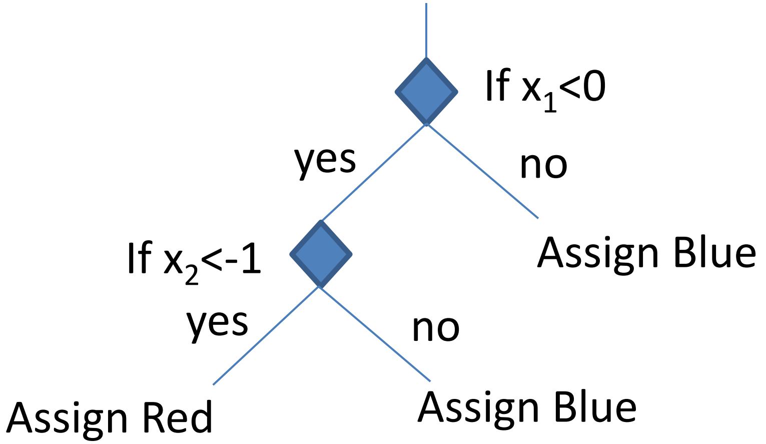
Continuing: Depth = 1



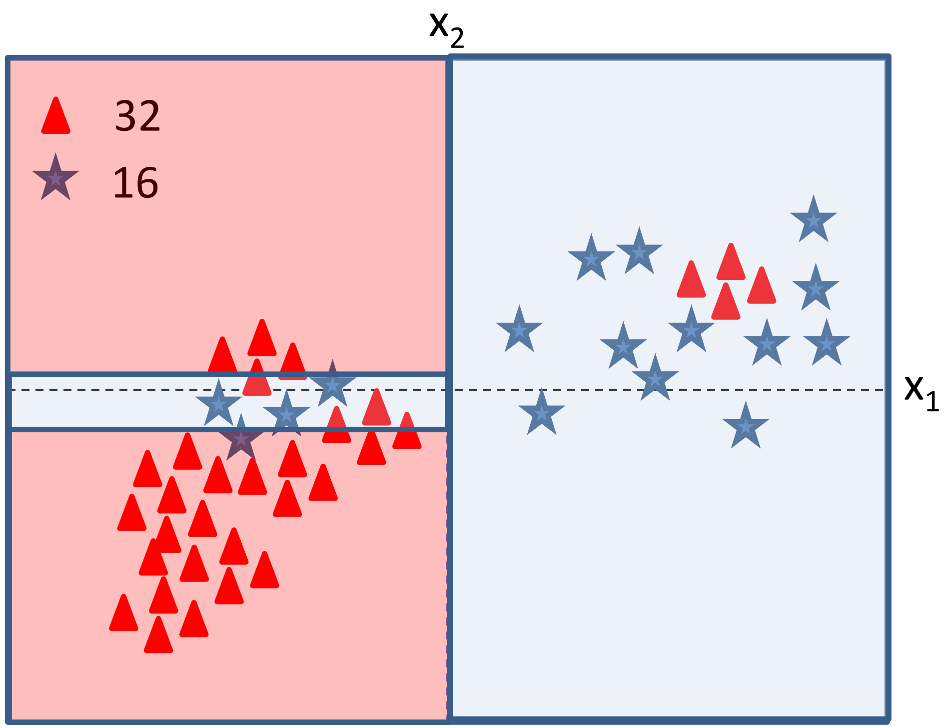


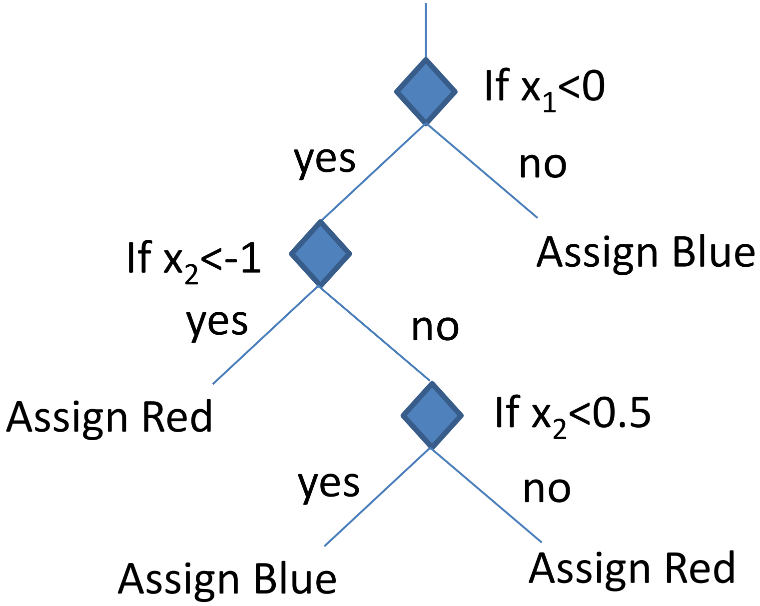
Continuing: Depth = 2



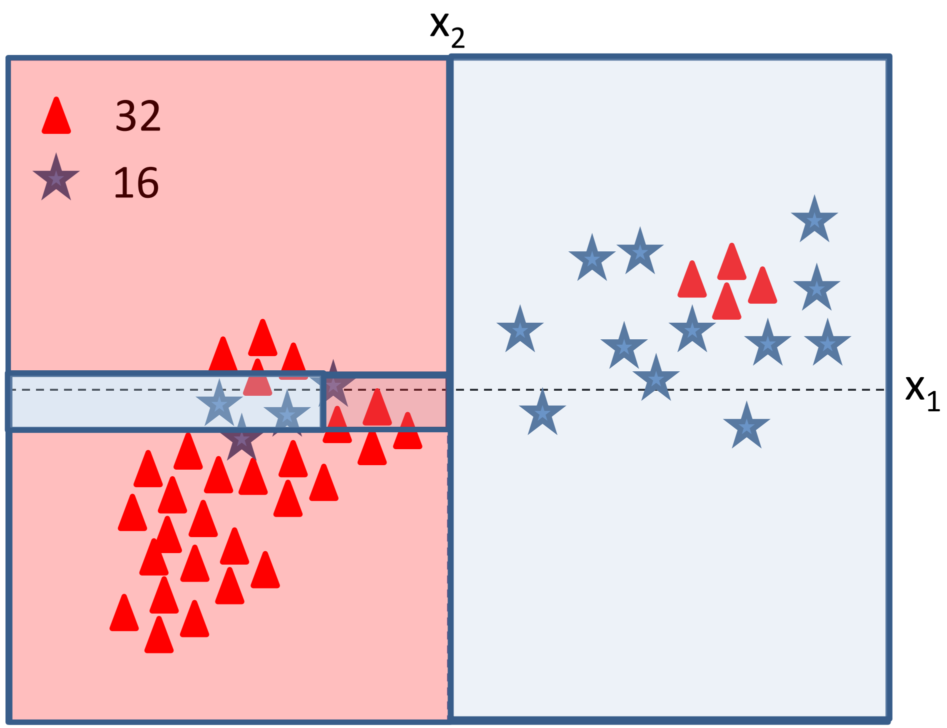


Continuing: Depth = 3

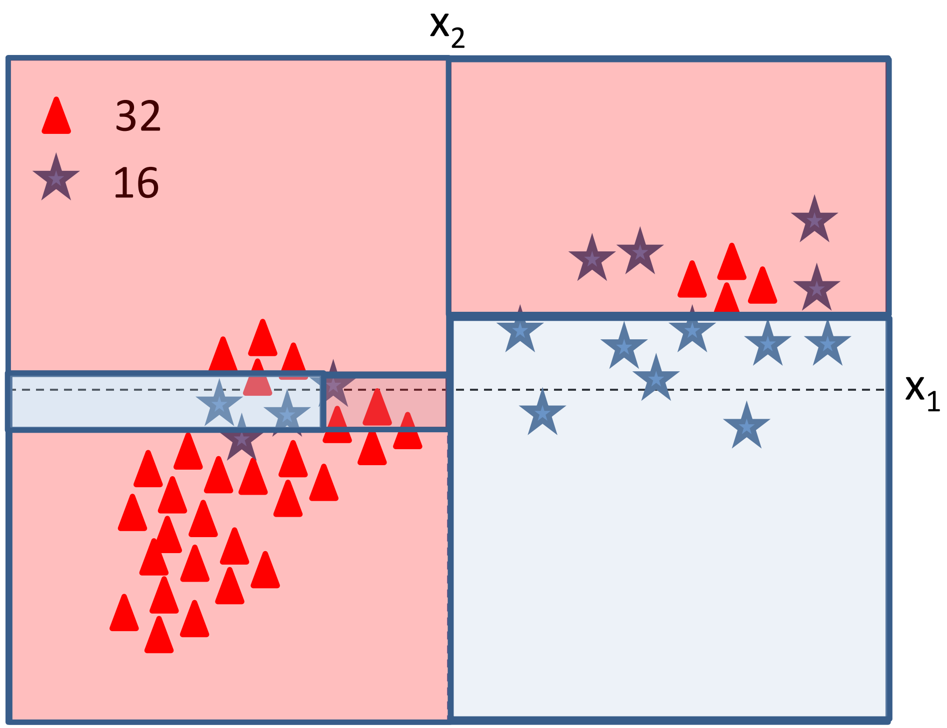




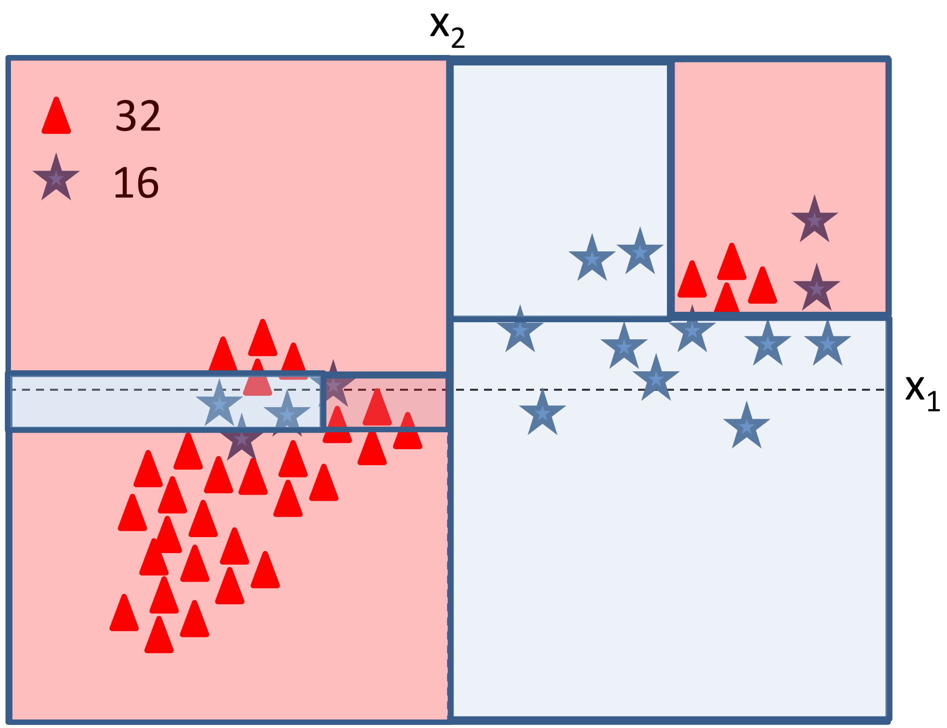
Continuing



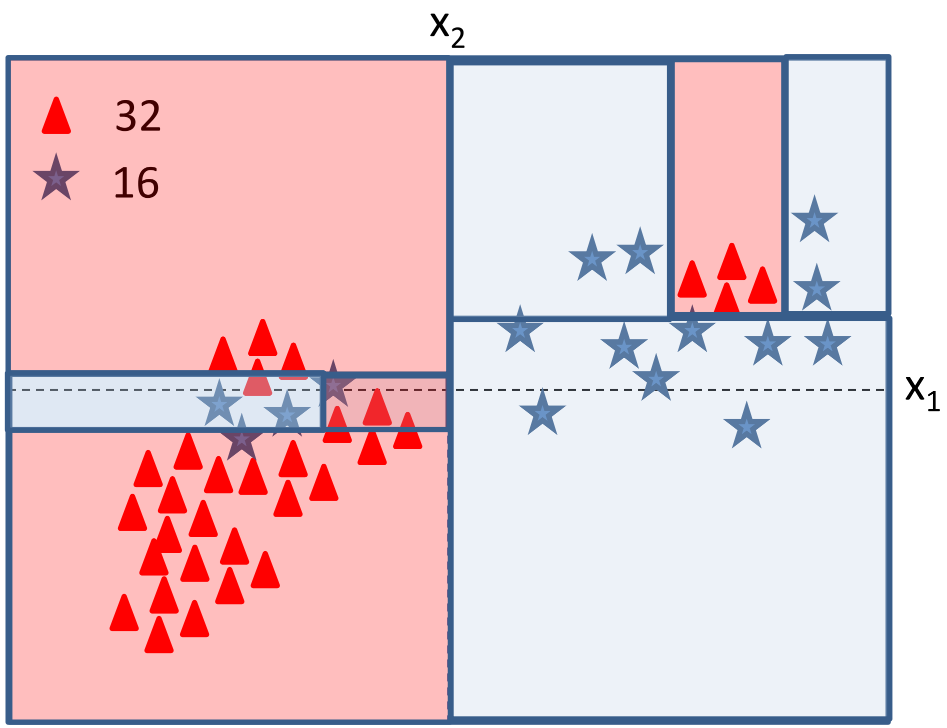
Continuing



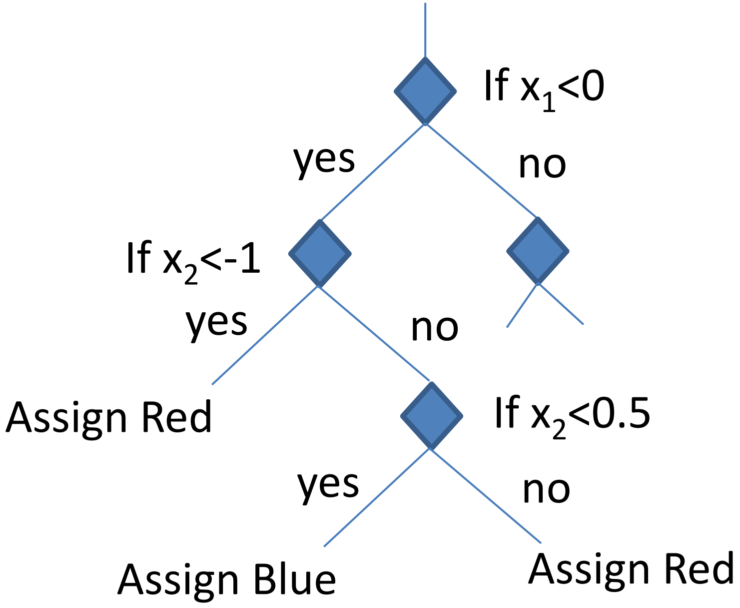
Continuing



Final



32
16
x_2
x_1

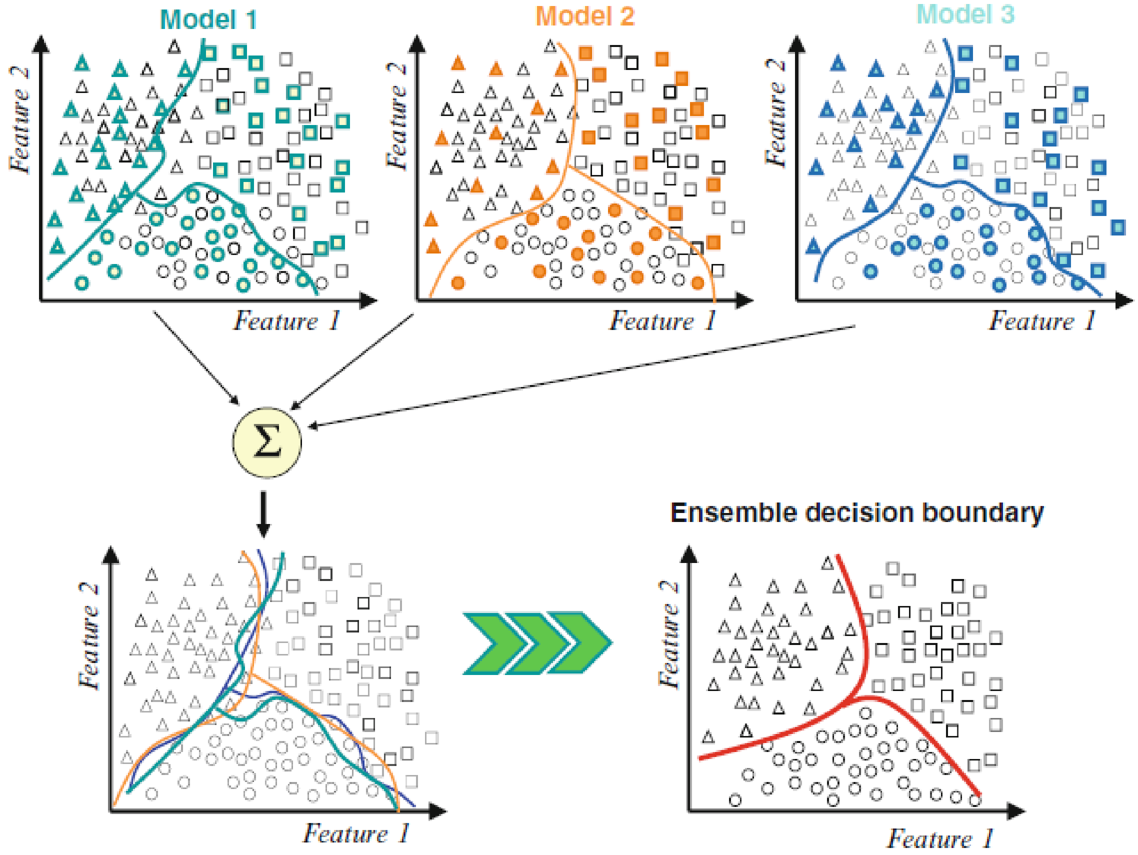


If x_1<0
yes
no
If x_2<-1
yes
no
Assign Red
If x_2<0.5
yes
no
Assign Blue
Assign Red

Tree: T

From Trees to Random Forests

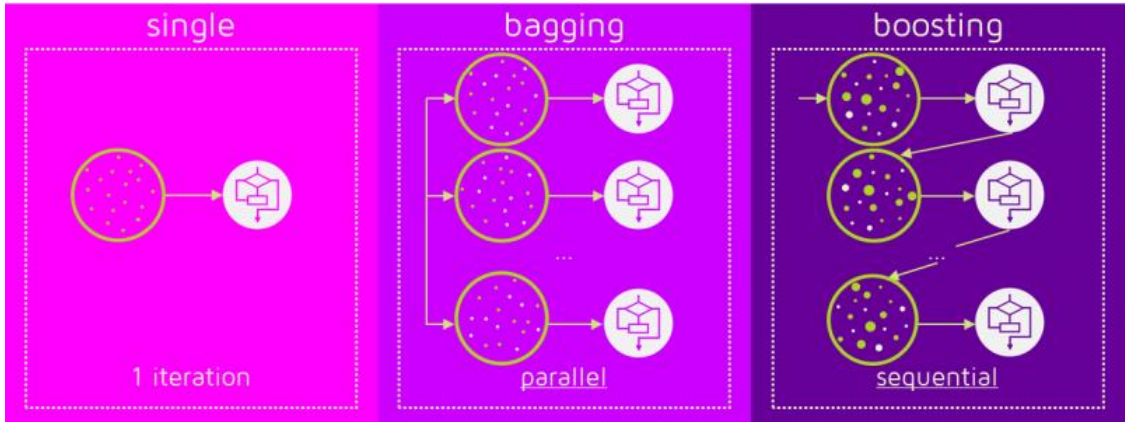
- Combine the predictions from multiple “weak” learners
  - Uncorrelated errors in predictions
    - Each learner makes errors on different examples
- How to make different classifiers
  - Different Data set partitioning
  - Different Features
  - Different parameters
  - Learning errors from previously trained methods



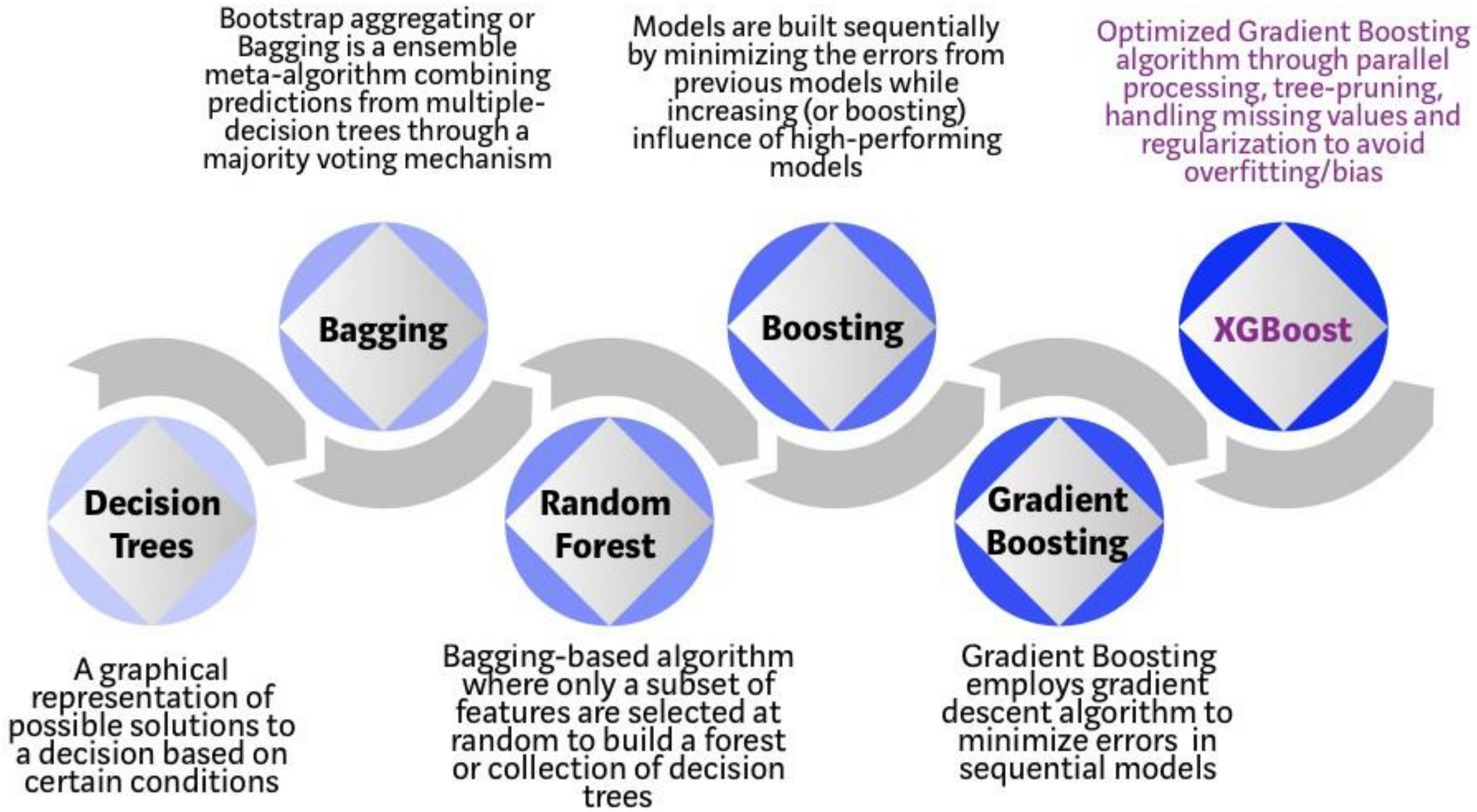
[Reading](#)
[Polikar 2006: http://users.rowan.edu/~polikar/RESEARCH/PUBLICATIONS/csm06.pdf](http://users.rowan.edu/~polikar/RESEARCH/PUBLICATIONS/csm06.pdf)
[Ensemble Machine Learning Methods and Applications (chapter 1), 2012](#)
https://doc.lagout.org/science/Artificial%20Intelligence/Machine%20learning/Ensemble%20Machine%20Learning_%20Methods%20and%20Applications%20%5BZhang%20%26%20Ma%202012-02-17%5D.pdf

From Trees to XGBoost

- Instead of "bagging" or boot-strap aggregating, i.e., combining "simple" trees by averaging, we can also combine them in series such that each tree "boosts" the decision of the previous one
  - Boosting involves incrementally building an ensemble by training each new model instance to emphasize the training instances that previous models mis-classified.



XGBoost

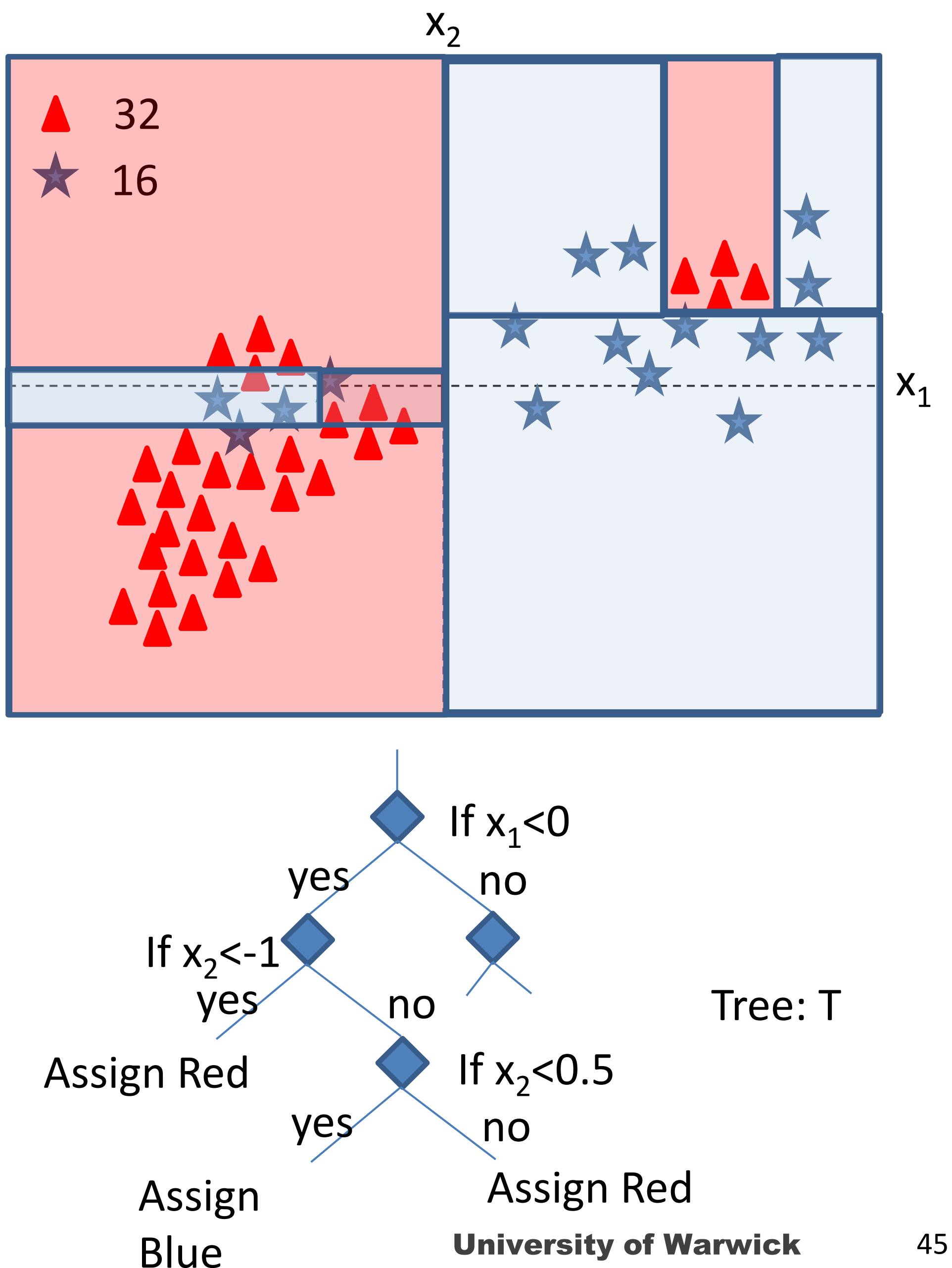

Bootstrap aggregating or Bagging is a ensemble meta-algorithm combining predictions from multiple-decision trees through a majority voting mechanism
Models are built sequentially by minimizing the errors from previous models while increasing (or boosting) influence of high-performing models
Optimized Gradient Boosting algorithm through parallel processing, tree-pruning, handling missing values and regularization to avoid overfitting/bias
Bagging
Boosting
XGBoost
Decision Trees
Random Forest
Gradient Boosting
A graphical representation of possible solutions to a decision based on certain conditions
Bagging-based algorithm where only a subset of features are selected at random to build a forest or collection of decision trees
Gradient Boosting employs gradient descent algorithm to minimize errors in sequential models

An REO/SRM View of Decision Trees

- Decision Trees, Random Forests, and Gradient Boosting Models
  - Representation
    - Output for a given input $\boldsymbol{x}$ is generated by following a tree structure
      - Can be modelled as: $f(\boldsymbol{x}; T)$ where $\boldsymbol{x}$ is an input and T is the representation of a specific tree
  - Evaluation
    - For a given tree structure, T, the model $f(\boldsymbol{x}; T)$ will generate a loss for each training example which can be minimized.
    - We can further penalize based on the structure of the tree itself. For example, if it is too complicated or if it uses too many features etc.

$$\min_T R(T) + \lambda \sum_i l(f(x_i; T), y_i)$$

  - Optimization
    - Can be done through gradient based optimization (as in XGBoost)
    - Or can go through an optimization process during the construction of the tree structure itself (e.g., by reduction of entropy or variance in each leaf)



An exercise into SRM

LEARNING TO RANK

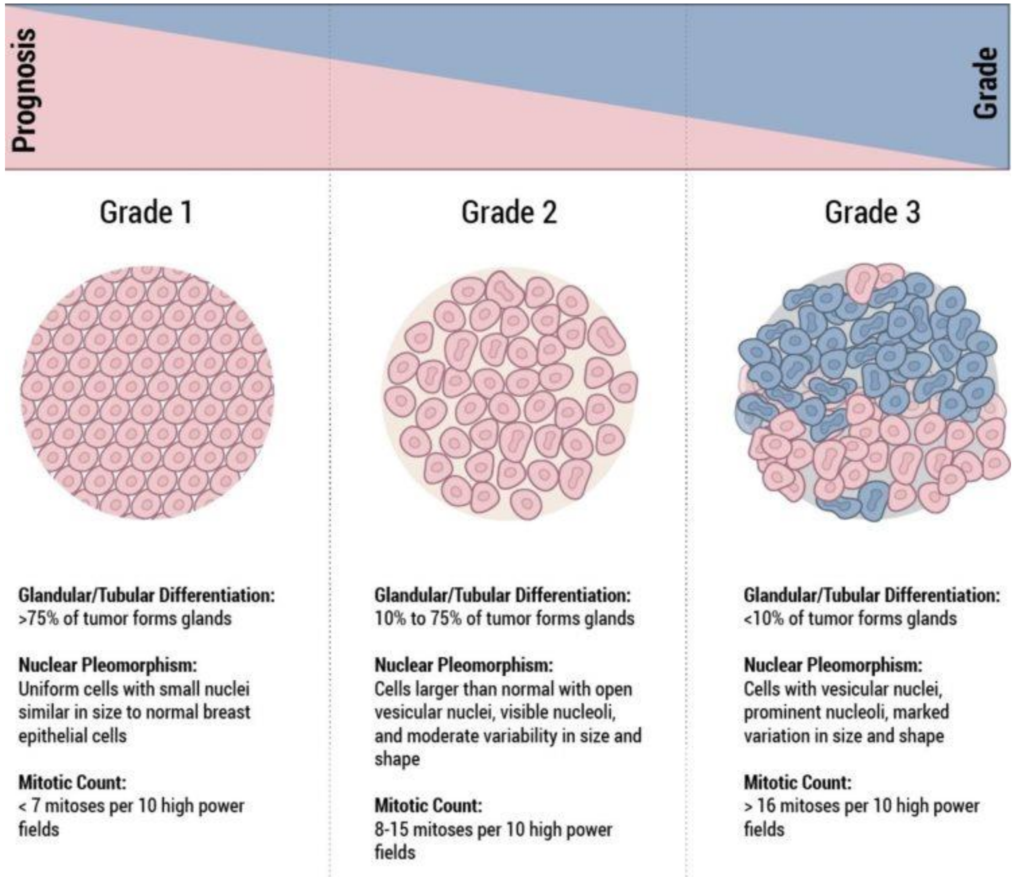
Learning to Rank

- Assign a rank to an input example
- Classification
  - Assign into classes
    - Typically no semantic relationship between classes (you cannot define greater than or less than in apples vs. oranges)
- Regression
  - Assign continuous variables
    - Age: 21 is greater than 18 (a relationship exists)
- Ranking
  - Assigning ranks
    - This is better or worse than that
  - Also called "Ordinal Regression"

Apple grading

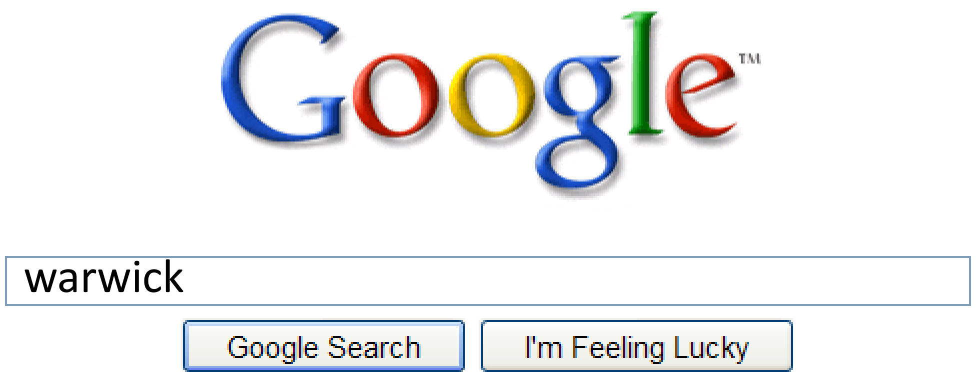
U.S. Extra Fancy
U.S. Fancy
U.S. No. 1
U.S. No. 1 Hail
U.S. Utility

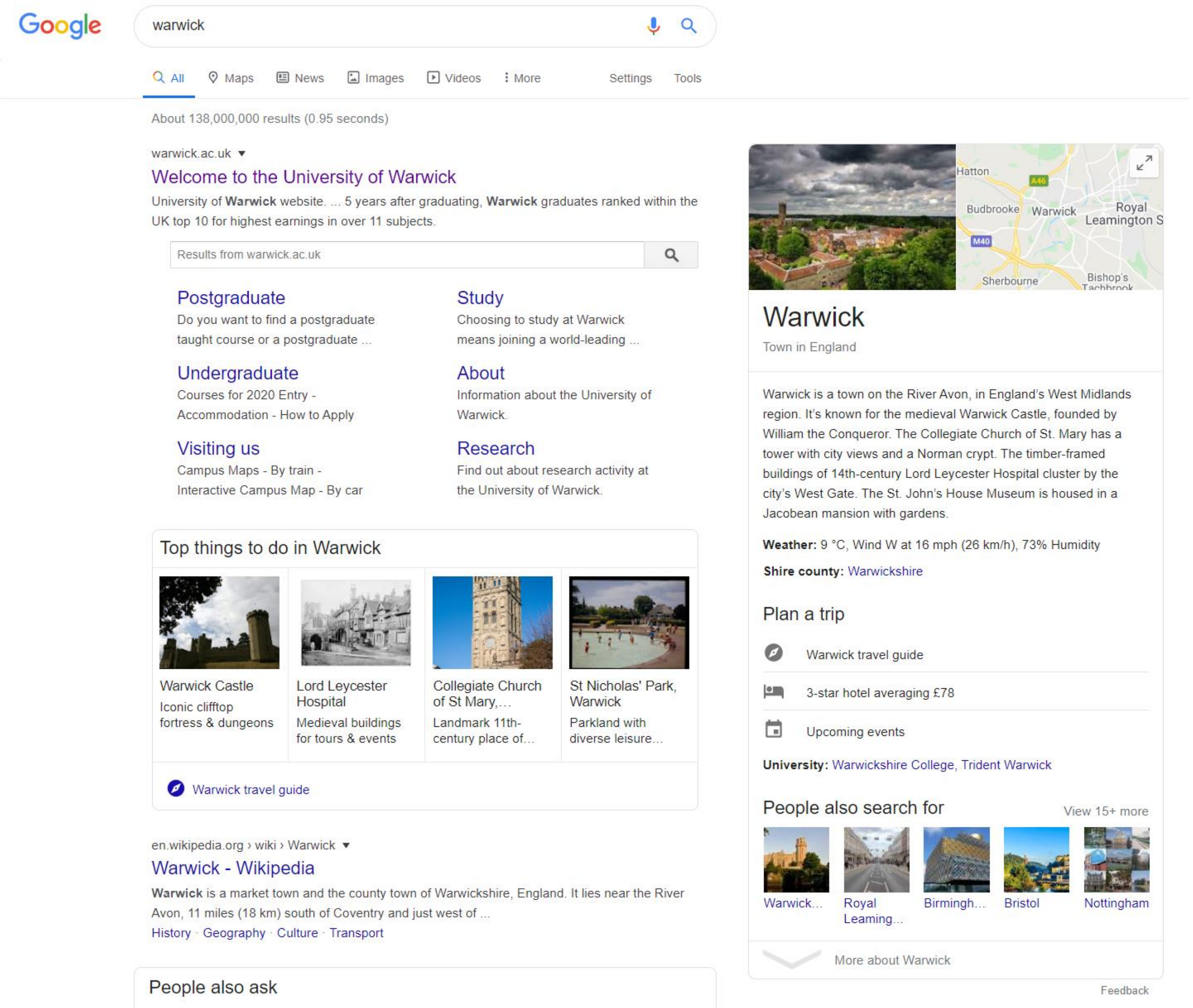
Cancer Grading



Ranking in Information Retrieval

- For a given query, a page that is more often clicked should be ranked higher than the one that isn't





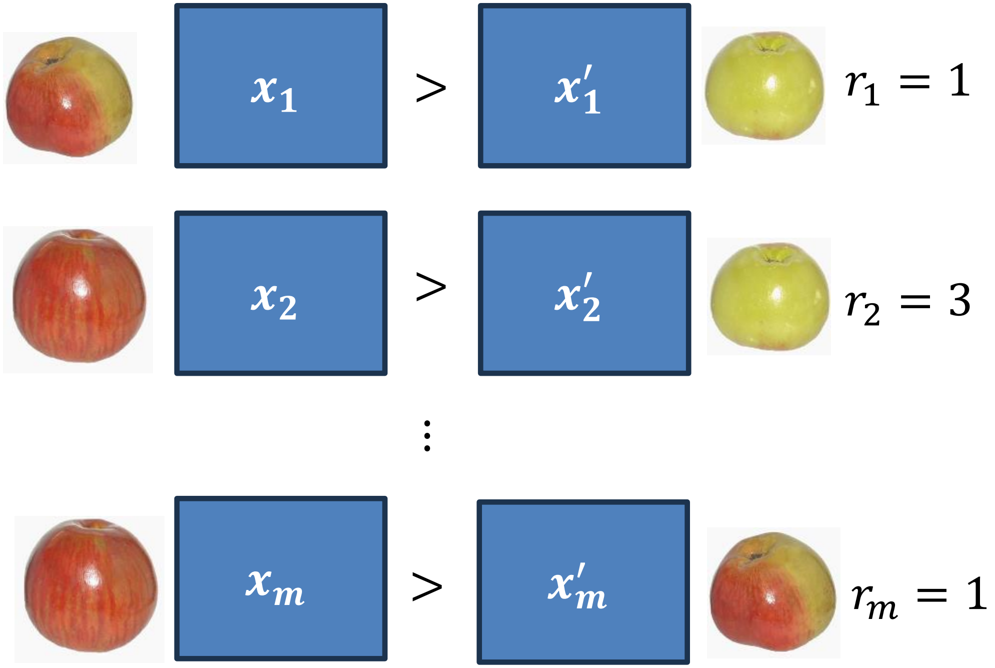
Learning to Rank

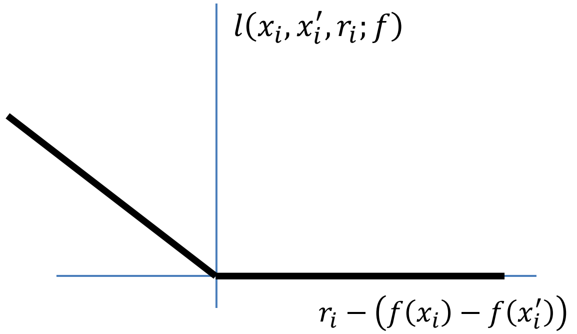
- **Problem Formulation**
  - Input: Training samples as pairs such that one $\boldsymbol{x_i}$ is to be ranked higher than another $\boldsymbol{x}_i'$ by an amount r_i
    - $S = \{(\boldsymbol{x_1}, \boldsymbol{x}'_1, r_1), \dots, (\boldsymbol{x_m}, \boldsymbol{x}'_m, r_m)\}$
      - Pairs of examples $(\boldsymbol{x}_i, \boldsymbol{x}'_i)$
      - Rank difference $r_i > 0$
    - Goal: Learn a function Ranking function: $f: X \rightarrow R$
- **Representation**: $f(\boldsymbol{x}; \boldsymbol{w})$
- **Evaluation**
  - A misranking will occur if $f(\boldsymbol{x_i}; \boldsymbol{w}) - f(\boldsymbol{x'}_i; \boldsymbol{w}) < r_i$
  - Thus, the loss becomes:

$$l(x_i, x_i', r_i; f) = \max\left(0, r_i - \left(f(\boldsymbol{x_i}) - f(\boldsymbol{x}_i')\right)\right)$$

  - Optimization problem becomes:

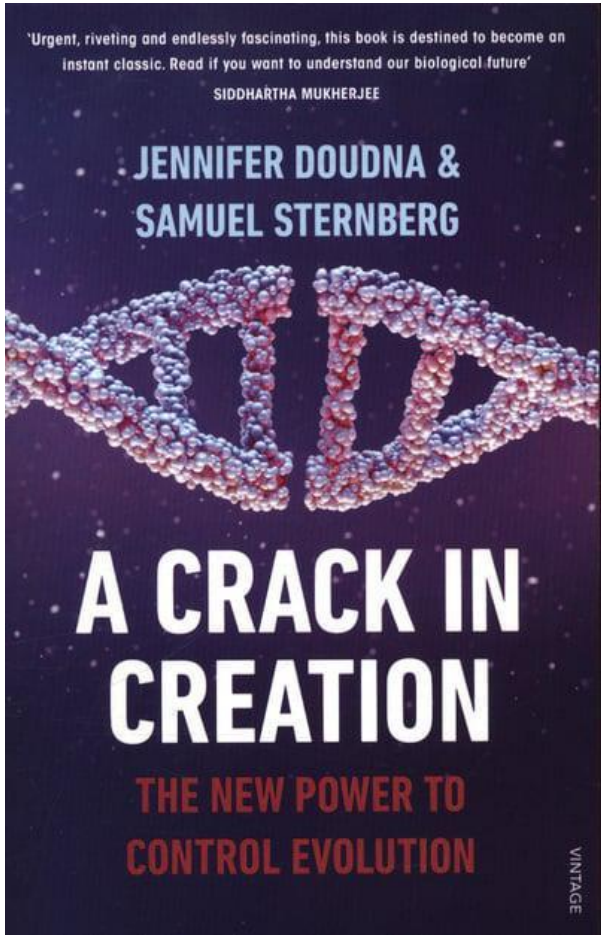
$$\min_w R(\boldsymbol{w}) + \sum_i l(x_i, x_i', r_i; f)$$





Predicting anti-CRISPR proteins

- Identify if a protein in a set of proteins is an Anti-CRISPR protein
  - Given: sets of sets of proteins (proteomes) in which at least one protein is an anti-CRISPR protein
    - Only 20 examples
  - Required: Rank the known anti-CRISPR protein higher than non-anti-CRISPR proteins in the proteome
- Used ranking constraints with an XGBoost model
- Able to identify new anti-CRISPR proteins in new species



Eitzinger, Simon, Amina Asif, Kyle E. Watters, Anthony T. Iavarone, Gavin J. Knott, Jennifer A. Doudna, and Fayyaz ul Amir Afsar Minhas. "Machine Learning Predicts New Anti-CRISPR Proteins." Nucleic Acids Research. April 16, 2020. https://doi.org/10.1093/nar/gkaa219

An exercise into SRM

RECOMMENDATION SYSTEMS

Recommendation Systems

- Task
  - Predict the rating or preference a user would give an item
- Given:
  - Training data: Matrix of Items rated by users
- Other names
  - Matrix Completion Problem
  - Information Filtering Problem

$$Y_{(M \times U)} = [y_{mu}]$$

MOVIE / USERS	Alice (1)	Bob (2)	Carol (3)	Dave (4)
Love at last	5	5	0	0
Romance Forever	5	**?**	**?**	0
Cute puppies of love	**?**	4	0	**?**
Nonstop car chases	0	0	5	4
Swords vs. Karate	0	0	5	**?**

https://help.netflix.com/en/node/100639
https://en.wikipedia.org/wiki/Recommender_system
Andrew Ng's lectures: https://www.youtube.com/playlist?list=PL_npY1DYXHPT-3dorG7Em6d18P4JRFDvH

REO/SRM for Collaborative Filtering Recommendations

- Representation
  - Represent each movie m by its features: $\boldsymbol{x}_m$
    - Note these features may not be known
  - Represent the prediction score for this movie for user u as: $\boldsymbol{w}_u^T\ \boldsymbol{x}_m$
    - Note that the weight vector $\boldsymbol{w}_{\boldsymbol{u}}$ characterizes each user
  - Whenever a user u rates a movie m, we set a flag s_{mu} to 1 otherwise 0
  - The rating for the movie m by user u is y_{mu}
- Evaluation
  - Prediction error for all labelled movies $$E = \frac{1}{2}\sum_{u=1}^{U}\sum_{m=1}^{M} s_{mu}(\boldsymbol{w}_u^T\ \boldsymbol{x}_m - y_{mu})^2$$
  - We add regularization terms: $\frac{\lambda_u}{2}\sum_{u=1}^{U}\|\boldsymbol{w}_u\|^2, \frac{\lambda_m}{2}\sum_{m=1}^{M}\|\boldsymbol{x}_m\|^2$
  - The complete optimization problem aims to find both user weights and movie features

$$\min_{\boldsymbol{w_u},\boldsymbol{x_m}} \frac{\lambda_u}{2}\sum_{u=1}^{U}\|\boldsymbol{w}_u\|^2 + \frac{\lambda_m}{2}\sum_{m=1}^{M}\|\boldsymbol{x}_m\|^2 + \frac{1}{2}\sum_{u=1}^{U}\sum_{m=1}^{M} s_{mu}(\boldsymbol{w}_u^T\ \boldsymbol{x}_m - y_{mu})^2$$

- Optimization
  - Can alternate between finding user weights and movie features

What can we do with this?

- We can rank the movies that were not ranked by a user
- We can also identify similar movies
    - Nearest neighbors over $\boldsymbol{x}^i$
- Or similar users
    - Nearest neighbors over $\boldsymbol{w}^j$
- Or identify popular trends of movies
    - – Average movie ratings across all users

An exercise into SRM

OTHER PROBLEMS

ML Task	ML Task
Classification (Binary and Multi-class: OVR, OVA, etc)	Out of Domain Detection
Regression	Novelty Detection/One-Class Classification
Dimensionality Reduction / Decomposition	Retrieval / Vector Database Search
Clustering	Prediction under domain shift or concept drift
Biclustering	Counterfactual prediction
Recommender System, Basket (item co-occurrence analysis)	Zero and Few Shot Prediction
Learning to Rank (Ordinal Regression)	Semi-Supervised Learning
Generative Modelling: Conditional and Unconditional	Weakly-supervised and multiple instance learning
Multi-task Prediction	Causal Learning, Inference and Discovery
Multi-Label Prediction	Active Learning
Survival Prediction (Churn Prediction or Failure Prediction)	Meta Learning
Adaptive Prediction Sets & Conformal Prediction	Curriculum Learning
Meta-Learning: Learning to learn and learning to optimize	Transfer Learning
Representation Learning	Contrastive and self-taught Learning
Open Set Recognition	Online and Continuous Learning
Subset Discovery	Reinforcement learning
Domain Specific tasks CV: Object detection, localization, counting, instance segmentation, semantic segmentation, image to image regression	Structured Output Learning Topic Modeling, Machine Translation, Counterfactual prediction Community discovery, graph learning, time series forecasting, …

“Nearly everything is really interesting if you go into it deeply enough.”

(Feynman)